This Page Is Inserted by IFW Operations and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

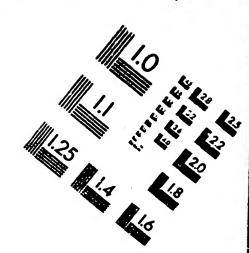
Defects in the images may include (but are not limited to):

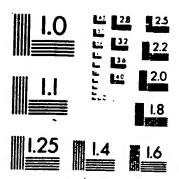
- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

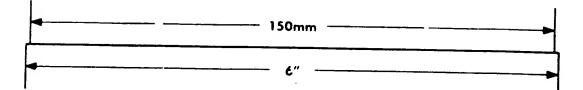
IMAGES ARE BEST AVAILABLE COPY.

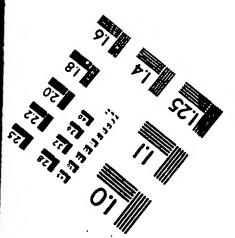
As rescanning documents will not correct images, please do not report the images to the Image Problem Mailbox.

IMAGE EVALUATION TEST TARGET (MT-3)

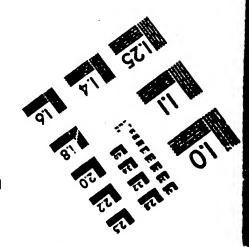








PHOTOGRAPHIC SCIENCES CORPORATIO! 1
770 BASKET ROAD
P.O. BOX 338
WEBSTER, NEW YORK 14580
(716) 265-1600



microunity

Zeus System Architecture

COPYRIGHT 1998 MICROUNITY SYSTEMS ENGINEERING, INC. ALL RIGHTS RUSERVED



Craig Hansen Chief Architect

MicroUnity Systems Engineering, Inc. 475 Potrero Avenue Sunnyvale, CA 94086.4118 Phone: 408.734.8100 Fax: 408.734.8136

email: craig@microunity.com http://www.microunity.com

Store

These operations add the contents of two registers to produce a virtual address, and store the contents of a register into memory.

Operation codes

Z'855	Store byte
S. 16.B	Store double big-endian
S.16AB	Store double aligned big-endian
2.16.L	Store double little-endian
S.16AL	Store double aligned little-endian
\$.32.6	Store quadlet big-endian
S.32.AB	Store quadlet aligned big-endian
\$.32.L	Store quadlet little-endian
S.32AL	Store quadlet aligned little-endian
S.64.B	Store octlet big-endian
S.64AB	Store octlet aligned big-endian
S.64.L	Store octlet little-endian
S.64AL	Store octiet aligned little-endian
S. 128.B	Store hexiet big-endian
S. 128AB	Store hexlet aligned big-endian
S.128.L	Store hexlet little-endian
S.128AL	Store hexlet aligned little-endian
S.MUX.64.A.B	Store multiplex octlet aligned big-endian
S.MUX.64.AL	Store multiplex octlet aligned little-endian

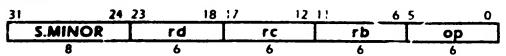
Selection

number format	P	size	alignment	ordering
byte	T	8	·	
integer		16 32 64 128		I. B
integer aligned		16 32 64 128	Α	L B
multiplex	MUX	64	Α	L B

Format

op rd,rc,rb

op(rd,rc,rb)



TS.8 need not specify byte ordering, not need it specify alignment checking, as it stores a single byte.

Description

An operand size, expressed in bytes, is specified by the instruction. A virtual address is computed from the sum of the contents of register re and the contents of register rh multiplied by operand size. The contents or register rd, treated as the size specified, is stored in memory using the specified byte order.

If alignment is specified, the computed virtual address must be aligned, that is, it must be an exact multiple of the size expressed in bytes. If the address is not aligned an "access disallowed by virtual address" exception occurs.

Definition

```
def Store(op.rd,rc,rt;) as
    case op of
         S8:
              size ← 8
         S16L S16AL S168, S16AB:
              size ← 16
         S32L S32AL S32B, S32AB:
              size ← 32
         S64L S64AL S64B, S64AB.
         SMUX64AB, SMUX64AL:
              size ← 64
         S128L S128AL, S128G, S128AB:
              ize ← 128
    endcase
    isze - logiszej
    case on of
        SB:
              order - undefined
         S16L S32L S64L S128L
         STEAL SEEAL SEEAL STEERL SMUXEEALE
             order ← L
        $16B, $32B, $64B, $128B,
         $16AB, $32AB, $64AB, $128AB, $MUX64ABI:
             order ← B
   Conditions
   c ← RegRead(rc. 64)
    b ← RegRead(rb. 64)
   VirtAddr + C + (b66-isse, C 11 Olsize-3)
   case op of
        S16AL S32AL S64AL S128AL
        $16AB, $32AB, $64AB, $128AB, -
        SMUX64AB, SMUX64AL:
             if f(bize-4.0 \neq 0) then
                  raise AccessDisallowedByVirtualAddress
        S16L S32L S64L S128L
        $16B, $32B, $64B, $128B:
        SB:
   endcase
   d ← RegRead(rd, 128)
   case op of
```

SB, \$16L, \$16AL, \$16B, \$16AB, \$32L, \$32AL, \$32B, \$32AB, \$64L, \$64AL, \$64B, \$64AB, \$128L, \$126AL, \$128B, \$128AB: \$tor-Memory(c,VirtAddr,size order,d_{\$12e-1..0}) \$MLIX64AB, \$MLIX64AI: lock

> a ← LoadMemoryW[c,VirtAddr,size,order] m ← [d_{127,64} & d_{63,0}] | [a & -d_{63,0}] StoreMemory[c,VirtAddr,size,order,m] endlock

endcase

enddef

Exceptions

Store Double Compare Swap

These operations compare two 64-bit values in a register against two 64-bit values read from two 64-bit memory locations, as specified by two 64-bit addresses in a register, and if equal, store two new 64-bit values from a register into the memory locations. The values read from memory are carenated and placed in a register.

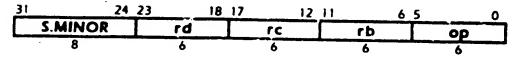
Operation codes

S.D.C.S.64.AB	Store double compare swap cottlet aligned big- endian
S.D.C.S.64.A.L	Store double compare swap octlet aligned little- endian

Format

op rd@rc.rb

rd=op(rd,rc,rb)



Description

Two virtual addresses are extracted from the low order bits of the contents of registers re and rb. Two 64-bit comparison values are extracted from the high order bits of the contents of registers re and rb. Two 64-bit replacement values are extracted from the contents of register rd. The contents of memory using the specified byte order are read from the specified addresses, treated as 64-bit values, compared against the specified comparison values, and if both read values are equal to the comparison values, the two replacement values are written to memory using the specified byte order. If either are unequal, no values are written to memory. The loaded values are catenated and placed in the register specified by rd.

The virtual addresses must be aligned, that is, it must be an exact multiple of the size expressed in bytes. If the address is not aligned an "access disallowed by virtual address" exception occurs.

Definition

def StoreDoubleCompareSwap(op.rd,rc,rb) as

size ← 64

Isize ← log(size)
case op of
SDCS64AL:
order ← 1
SDCS64AB:

```
order ← B
     endcase
     c ← RegRead(rc. 128)
     b ← RegRead(rb, 128)
     d ← RegRead(rd, 128)
     if (c_{2..0} \neq 0) or (b_{2..0} \neq 0) then
           raise AccessDisallowedByVirtualAddress
     endil
     lock
           a \leftarrow LoadMemoryW[c_{63..0},c_{63..0},64,order] \quad I \quad Lo.:dMemoryW[b_{63..0},b_{63..0},64,order]
           d ((c127.64 | 1 b127.64) = a) then
                StoreMemory([c63.0,c63.0.64,order,d127.64]
                StoreMemory[b63.0,b63.0.64,order,d63.0]
           endil
     endlock
     RegWritefro. 128, a)
enddef
```

Exceptions

Access disallowed by varual address Access disallowed by tag Access disallowed by global TB Access disallowed by local TB Access detail required by tag Access detail required by local TB Access detail required by global TB Access detail required by global TB Local TB miss

Store Immediate

These operations add the contents of a register to a sign-extended immediate value to produce a virtual st. dress, and store the contents of a register into memory.

Operation codes

Name and Address of the Owner, where the Owner, which is the Owne	والمراجع والمناب والمنطور والمناب والمناب والمناب والمراجع والمناب والمناب والمناب والمناب والمناب والمناب والمناب
S.I.8 ²³	Store immediate byte
S.I.16.A.B	Store immediate double aligned big-endian
S.I. 16.B	Store immediate double big-endian
S.I. 16.AL	Store immediate double aligned little-endian
S.I. 16.L	Store immediate double little-endian
S.I.32 A.B	Store immediate quadlet aligned big-endian
S.I.32.B	Store immediate quadlet big-endian
S.I.32.A.L	Store immediate quadlet aligned tittle-endian
S.I.32.L	Store immediate quadlet little-endian
S.I.64.A.B	Store immediate octlet aligned big-endian
S.I.64.B	Store immediate octlet big-endian
S.I.64.A.L	Store immediate octlet aligned little-endian
S.I.64.L	Store immediate octlet little-endian
S.I.128 AB	Store immediate hexlet aligned big-endian
S.I. 128.B	Store immediate hexlet big-endian
S.I. 128.A.L	Store immediate hexlet aligned little-endian
S.I. 128.L	Store immediate hextet little-endian
S.MUXI.64.A.B	Store multiplex immediate octlet aligned big-endian
S.MUXI.64.A.L	Store multiplex immediate octlet aligned little-endian

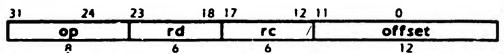
Selection

number format	ορ	size	alignment	orde	ring
byte		8			
integer		16 32 64 128		L	В
integer aligned		16 32 64 128	Α	L	R
multiplex	MUX	54	Α	L	В

Format

S.op.I.size.align.order rd,rc,offset

sopisizealignorder(rd,rc,offset)



^{21/}S1.8 need not specify byte ordering, nor need it specify alignment checking, as it stores a single byte.

Description

An operand size, expressed in bytes, is specified by the instruction. A virtual address is computed from the sum of the contents of register re and the sign-extended value of the offset field, multiplied by the operand size. The contents of register rd, treated as the size specified, are written to memory using the specified byte order.

The computed virtual address must be aligned, that is, it must be an exact multiple of the size expressed in bytes. If the address is not aligned an "access disallowed by virtual address" exception occurs.

Definition

```
del Storelmmediate(op,rd,rc,offset) as
    case op of
         S#8:
              size ← 8
         SHOLL SHOAL SHORE SHOAR
              size - 16
         SI32L SI32AL SI32B, SI32AB
              size ← 32
         SIGAL SIGAAL SIGAB, SIGAAB, SMUXIGAAB, SMI DOGAAL:
              size - 64
         S1128L S1128AL S1128AL S1128AL
              size ← 128
    endcase
    isize - logisize
    case op of
        S18:
             order - undefined
        SI16L SI32L SI64L SI128L
        SITEAL SIZZAL SIEGAL SITZBAL SMUXIEGAL:
             order ← L
        S116B, S132B, S164B, S1128B.
        SI16AB, SI32AB, SI64AB, SI128AB, SMUXI64AB:
             order ← B
   endcase
   c ← RegRead(rc, 64)
   VirtAddr - c + foffset $ 5-1812e | 1 offset | 1 Ofsize-31
   case op of
        SITEAL SIZZAL SIEGAL SITZBAL
        SI16AB, SI32AB, SI64AB, SI128AB,
        SMUXI64AB, SMUXI64AL:
             # Kruze-4 0 # 0 then
                  raise AccessDisallowedByVirtualAddress
        S116L S132L S164L S1128L
        S116B, S132B, S164B, S1128B:
        SIB:
   endcase
   d ← RegReadird 128
   case op of
       SIB.
        SITEL SITEAL SITEB, SITEAB,
       SI32L SI32AL SI32B, SI32AB
```

SIGAL, SIGAAL, SIGAB, SIGAAB, SI128L, SI128AL, SI128B, SI128AB; StoreMemory(c,ViriAddr,size,order,d_{size-1_0}) SMUXIGAAB, SMUXIGAAL; lock

> a ← LoadMemoryW/c,VirtAddr,size,order) m ← [d_{127_64} & d_{63_0}] | [a & -d_{63_0}] StoreMemory/c,VirtAddr,size,order,m] endlock

endcase enddef

Exceptions

Access deallowed by various address
Access deallowed by tag
Access deallowed by global TB
Access deallowed by local TB
Access detail required by tag
Access detail required by global TB
Access detail required by global TB
Local TB mass
Global TB mass

Store Immediate Inplace

These operations add the contents of a register to a sign-extended immediate value to produce a virtual address, and store the contents of a register into memory.

Operation codes

SAS.1.64AB	Store add swap immediate octlet aligned big endian
SAS.1.64AL	Store add swap immediate octlet aligned little-endian
S.C.S.I.64.A.B	Store compare swap immediate notifet aligned big-endian
S C.S.I.64 AL	Store compare swap immediate uctlet aligned little-endian
S.M.S.1.64.A.B	Store multiplex swap immediate octlet aligned big-endian
S.M.S.1.64.A.L	Store multiplex swap immediate octlet aligned little-endiar:

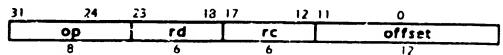
Selection

number format	lop	size	alignment	orde	erina
add-swap	AS	64		I	R
compare-swap	ट	64	A	1	R
multiplex-swap	MD	64		1	R

Format

S.op.I.64.align.order rd@rc.offset

rd=sopi64alignorder(rd,rc,offset)



Description

A virtual address is computed from the sum of the contents of register re and the sign extended value of the offset field. The contents of memory using the specified byte order are read and treated as a 64-bit value. A specified operation is performed between the memory contents and the original contents of register rd, and the result is written to memory using the specified byte order. The original memory contents are placed into register rd.

The compated virtual address must be aligned, that is, it must be an exact multiple of the size expressed in bytes. If the address is not aligned an "access disallowed by virtual address" exception occurs.

Definition

def StoreimmediateInplaceJop,id,rc,offsetj as

size ← 64 !size ← log(size) case op of

```
SASIGNAL SCSIGNAL SMSIGNAL:
              order - L
         SASI64AB, SCSI64AB, SMSI64AB-
              order ← B
    endcase
    c - Regiteadire, 64)
    VirtAddr ← c + foffset $5-tsize | 1 offset | 1 Offset-3|
    d (chize-4.0 \neq 0 then
         raise AccessDisallowedByVirtualAddress
    endf
    d - RegReadtrd, 128)
    case op of
         SASIGNAL SASIGNAL:
              · lock
                    a - LoadMemoryWk,VirtAddr,size,order)
                    StoreMemoryle, VirtAcide, size, order, de 3 ordi
               enJlack
         SCSIGAAR SCSIGAAL:
               lock
                    a -- LoadMemoryWk 'VirtAddr.size,order)
                    # la = do; of then
                         StoreMemory(c, VirtAddr, size, order, d<sub>127.64</sub>)
               endlock
         SMSI64AB, SMSI64AL:
               lock
                    a ← LoadMemoryWk,VirtAcdr,size,order)
                    m - (d127 64 & d63.0) 1 ia & -d63 of
                    StoreMemoryk:, VirtAddr., rize, order, mj
               enchack
     endcase
     RegWritefrd, 64, al
enddel
```

Exceptions

Access disallowed by virtual address Access disallowed by tag Access disallowed by global TB Access disallowed by local TB access detail required by local TB Access detail required by global TB Local TB miss.

Store Inplace

These operations add the contents of two registers to produce a virtual address, and store the contents of a register into memory.

Operation codes

SAS.64AB	Store add swap octlet aligned big-endian			
SAS.64AL	Store add swap octlet aligned little-endian			
S.C.S.64.AB	Store compare swap octlet aligned big-endian			
S.C.S.64AL	Store compare swap octlet aligned little-endian			
SMS.64AB	Store multiplex swap octlet aligned big-endian			
SMS.64AL	Store multiplex swap octlet aligned little-endian			

Sejection

number format	ОР	size	alignment	ordering
add-swao	AS	64	Α	L B
compare-swap	C.S	64		L B
multiplex-swap	M.S	64	Α	L E

Format

op rdercrb

rd=op(rd,rc,rb)

31	24 23	11	8 17	12 11	6 5	. 0
S.MI	NOR	rd	rc	-	b	ор
	ρ	6	6		6	6

Description

A virtual address is computed from the sum of the contents of register re and the contents of register rb multiplied by operand size. The contents of memory using the specified byte order are read and treated as 64 bits. A specified operation is performed between the memory contents and the original contents of register rd, and the result is written to memory using the specified byte order. The original memory contents are placed into register rd.

The computed virtual address must be aligned, that is, it must be an exact multiple of the size expressed in bytes. If the address is not augned an "access disallowed by virtual address" exception occurs.

Definition

del Storeinplace(op,rd,rc,rb) as size ← 64 isize ← logiuze) case op of

```
SASGAAL, SCSGAAL, SMSGAAL:
                order - L
           SASGAAB, SCSGAAB, SANGAAB.
                order - B
     ervicase
     c - RegReadire, 64)
     b - RegReadirb, 64)
     Virthdor + c + (Dec-sure 0 11 0/128-3)
     # Kruze-4 0 = 0 then
           raise AccessDisallowedByWirtualAddress
     endel
     d - PegReadird, 128
     case op of
           SASSAAB, SASSAAL
                lock
                     a + LoadMemoryWjc,VirtAddr,size,ordei)
                     StoreMemorytr. WrtAddr, size, order, d<sub>6.3</sub> 0+al
               expluct
          SCS64AB SCS64AL
               lock
                     a - LoadNerncsyW/c.VirtAddr.size.order)
                    # 10 . de3 of then
                         "toreMemoryk, VirtAddr, size, order, d | 27, 64)
                     andil
               endiate
          SASSAAB, SASSAAL. *
               toct.

→ LoadMemoryWk; VirtAdd: size, order)

                    m - 1012/ 64 6 063 of 1 fa 6 -063 of
                    StoreMemoryte, WtAddr.size, order, mj
               endlock
     endcase
     Regultizeted, 64, at
enddel
```

Exceptions

Access disallowed in virtual address Access disallowed in fig. Access disallowed in global TH Access disallowed in local TH Access detail required in global TH Local TH mass.

Group Add

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

	ومرور والمراجع والمراجع والمتحار
GADD.8	Group add bytes
GADD.16	Group add doublets
GADC 32	Group add quadlets
GADD.64	Group add octlets
GADD.128	Group add hexlet
G.ADD.L8	Group add limit signed bytes
GADD.L16	Group add limit signed doublets
GADD.L32	Group add limit signed quadlets
GADD.L.64	Group add limit signed octlets
GADD.L 128	Group add limit signed nextet
GADD.LU.8	Group add limit unsigned bytes
GACOLU.16	Group add Irmit unsigned doublets
GADD.I.U.32	Group add limit unsigned quadlets
GADD.LU.64	Group add limit unsigned octlets
GADO.LU.128	Group add limit unsigned hexlet
GADD.8.O	Group add signed bytes check overflow
GADD.16.O	Group add signed doublets check overflow
GADD.32.O	Group add signed quadlets check overflow
GADD.64.O	Group add signed octlets check overflow
GADD.128.O	Group add signed hexlet check overflow
GADD.U.8.O	Group add unsigned bytes check overflow
GADD.U.16 O	Group add unsigned doublets check overflow
GACO.U.32 O	Group add unsigned quadlets check overflow
GADD.U.64.O	Group add unsigned octicts check overflow
G.ADD.U.128.O	Group add unsigned hexlet check overflow

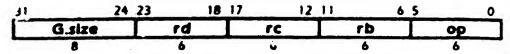
Redundancies

GADD.size rd=rc,rc	⇔	G.SHL.I.size rd=rc,1
GADU.size.O rd=rc,rc	⇔	G.SHL1.size.O rd=rc,1
GADD.U.size.O rd=rc.rc	⇔	G.SHL.I.U.size.O rd=rc, 1

Format

G.op.sizerJ=rc.rb

rd=gopsize(rc.rb)



Description

The contents of registers re and rb are partitioned into groups of operands of the size specified and added, and if specified, checked for overflow or limited, yielding a group of results, each of which is the size specified. The group of results is catenated and placed in register rd.

Definition

```
del Groupinp.size,rd.rc.rbj
    c - RegReacirc, 128|
    b - RegReadirb, 128
    case op of
         GAW.
              for i - 0 to 128-size by size
                    i esterili e (materili e proserili
         GADOL:
              for i ← 0 to 128-size by size
                   1 ← Knizeri II Chazeri d. * (Diogram II Diogram d
                   Support : ← Rare # Sare 1) ? Rare 11 [Me.] . Sare-10
              endfor
         GADO LU
              for i ← 0 to 128-size by size
                   1 +- 101 11 Cropper 1 + 101 11 Dropper 1
                   Source : -- Rare # 01 7 (1126) lure 1 0
              endfor
         GACOO
              for 1 - 0 to 128-size by size
                   1 - Knoppe 1 11 Conspect of a (Donate) 11 Donate 1 of
                   if take # take | then
                        raise FixedPointAnthmetic
                   Juste I'' ← pate 1 0
              enulfor
         G.ADD.U.O:
              for i ← 0 to 128-size by size
                   t ← (C) | Course | d + (C) | | Dougse | d
                   # taze # 0 then
                        raise fixedForntAnthimetic
                   Augret 1 + isre-1 0
```

endfor

endcase RegWritefrd, 128, aj enddef

Exceptions

fixed point anthinetic

Group Add Halve

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

GADD.H.8.C Group add halve signed bytes floor GADD.H.8.F Group add halve signed bytes floor GADD.H.8.N Group add halve signed bytes nearest GADD.H.8.Z Group add halve signed bytes zero GADD.H.8.Z Group add halve signed doublets ceiling GADD.H.16.C Group add halve signed doublets rearest GADD.H.16.N Group add halve signed doublets rearest GADD.H.16.N Group add halve signed doublets zero GADD.H.16.Z Group add halve signed doublets rearest GADD.H.16.Z Group add halve signed quadlets ceiling GADD.H.32.C Group add halve signed quadlets ceiling GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.N Group add halve signed quadlets rearest GADD.H.32.N Group add halve signed octlets floor GADD.H.64.C Group add halve signed octlets floor GADD.H.64.C Group add halve signed octlets floor GADD.H.64.P Group add halve signed octlets floor GADD.H.64.N Group add halve signed octlets rearest GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.C Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet floor GADD.H.128.Z Group add halve signed hexlet nearest GADD.H.128.Z Group add halve unsigned bytes floor GADD.H.128.Z Group add halve unsigned bytes floor GADD.H.128.F Group add halve unsigned bytes floor GADD.H.128.F Group add halve unsigned doublets ceiling GADD.H.128.N Group add halve unsigned doublets ceiling GADD.H.128.N Group add halve unsigned doublets ceiling GADD.H.128.N Group add halve unsigned doublets ceiling GADD.H.128.C Group add halve unsigned doublets floor GADD.H.128.C Group add halve unsigned doublets floor GADD.H.128.C Group add halve unsigned doublets ceiling GADD.H.128.C Group add halve unsigned doublets ceiling GADD.H.128.C Group add halve unsigned octlets nearest GADD.H.128.C Group add halve unsigned octlets floor GADD.H.128.C Group add halve unsigned octlets nearest GADD.H.128.C Group add halve unsigned hexlet ceiling GADD.H.128.C Group add halve unsigned hexlet ceiling GADD.H.128.C Group add halve unsigned hexlet ceiling GADD.H.128.R Group add halve unsigned hexlet		
GADD.H.8.N Group add halve signed bytes nearest GADD.H.16.C Group add halve signed doublets ceiling GADD.H.16.F Group add halve signed doublets floor GADD.H.16.N Group add halve signed doublets nearest GADD.H.16.N Group add halve signed doublets nearest GADD.H.16.N Group add halve signed doublets nearest GADD.H.32.C Group add halve signed doublets zero GADD.H.32.F Group add halve signed quadlets ceiling GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed octlets ceiling GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.P Group add halve signed octlets rearest GADD.H.64.Z Group add halve signed octlets nearest GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.N Group add halve signed hexlet rearest GADD.H.128.N Group add halve signed hexlet rearest GADD.H.128.N Group add halve unsigned bytes floor GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned doublets rearest GADD.H.U.8.N Group add halve unsigned doublets rearest GADD.H.U.8.N Group add halve unsigned doublets rearest GADD.H.U.16.F Group add halve unsigned doublets rearest GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.F Group add halve unsigned doublets rearest GADD.H.U.3.C Group add halve unsigned doublets floor GADD.H.U.3.C Group add halve unsigned doublets rearest GADD.H.U.3.C Group add halve unsigned doublets rearest GADD.H.U.3.C Group add halve unsigned octlets floor GADD.H.U.3.C Group add halve unsigned octlets rearest GADD.H.U.6.4.F Group add halve unsigned octlets rearest GADD.H.U.6.4.F Group add halve unsigned octlets floor GADD.H.U.6.4.F Group add halve unsigned octlets floor GADD.H.U.6.8.F Group add halve unsigned octlets floor	GADD H.8.C	Group add halve signed bytes ceiling
GADD.H.8.2 Group add halve signed bytes zero GADD.H.16.C Group add halve signed doublets ceiling GADD.H.16.F Group add halve signed doublets floor GADD.H.16.N Group add halve signed doublets nearest GADD.H.16.Z Group add halve signed doublets zero GADD.H.16.Z Group add halve signed doublets zero GADD.H.32.C Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.F Group add halve signed octlets floor GADD.H.64.P Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets zero GADD.H.128.F Group add halve signed hexlet ceiling GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.128.Z Group add halve unsigned bytes floor GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned doublets rearest GADD.H.U.8.N Group add halve unsigned doublets rearest GADD.H.U.16.C Group add halve unsigned doublets rearest GADD.H.U.16.N Group add halve unsigned doublets rearest GADD.H.U.3.2.N Group add halve unsigned octlets floor GADD.H.U.3.2.N Group add halve unsigned octlets floor GADD.H.U.6.4.F Group add halve unsigned octlets floor GADD.H.U.6.4.F Group add halve unsigned octlets floor GADD.H.U.6.4.N Group add halve unsigned hexlet ceiling GADD.H.U.6.8.N Group add halve unsigned octlets floor		Group add halve signed bytes floor
GADD.H.16.C Group add halve signed doublets ceiling GADD.H.16.F Group add halve signed doublets floor GADD.H.16.N Group add halve signed doublets nearest GADD.H.16.Z Group add halve signed doublets zero GADD.H.32.C Group add halve signed quadlets ceiling GADD.H.32.F Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.P Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets zero GADD.H.128.C Group add halve signed hexlet floor GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet nearest GADD.H.128.Z Group add halve unsigned bytes floor GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.8.N Group add halve unsigned doublets ceiling GADD.H.U.8.N Group add halve unsigned doublets floor GADD.H.U.8.N Group add halve unsigned doublets floor GADD.H.U.3.Z Group add halve unsigned quadlets floor GADD.H.U.3.Z Group add halve unsigned quadlets floor GADD.H.U.3.Z Group add halve unsigned octlets floor GADD.H.U.3.Z Group add halve unsigned octlets floor GADD.H.U.3.Z Group add halve unsigned octlets floor GADD.H.U.6.4 Group add halve unsigned octlets floor		Group add halve signed bytes nearest
GADD.H.16.F Group add have signed doublets floor GADD.H.16.N Group add have signed doublets nearest GADD.H.16.Z Group add have signed doublets zero GADD.H.32.C Group add have signed quadlets ceiling GADD.H.32.F Group add have signed quadlets floor GADD.H.32.N Group add have signed quadlets nearest GADD.H.32.Z Group add have signed quadlets zero GADD.H.64.C Group add have signed octlets ceiling GADD.H.64.F Group add have signed octlets floor GADD.H.64.N Group add have signed octlets nearest GADD.H.128.C Group add have signed octlets zero GADD.H.128.F Group add have signed hexlet ceiling GADD.H.128.N Group add have signed hexlet floor GADD.H.128.N Group add have signed hexlet nearest GADD.H.128.Z Group add have signed hexlet zero GADD.H.128.Z Group add have signed hexlet zero GADD.H.128.C Group add have unsigned bytes floor GADD.H.128.C Group add have unsigned bytes floor GADD.H.128.C Group add have unsigned bytes nearest GADD.H.128.C Group add have unsigned doublets ceiling GADD.H.128.N Group add have unsigned doublets rearest GADD.H.128.N Group add have unsigned doublets floor GADD.H.128.N Group add have unsigned doublets rearest GADD.H.128.N Group add have unsigned doublets floor GADD.H.128.N Group add have unsigned doublets rearest GADD.H.128.C Group add have unsigned doublets rearest GADD.H.128.C Group add have unsigned doublets floor GADD.H.128.C Group add have unsigned doublets rearest GADD.H.128.C Group add have unsigned doublets rearest GADD.H.128.C Group add have unsigned doublets rearest GADD.H.128.C Group add have unsigned octlets rearest	GADD.H.8.Z	Group add halve signed bytes zero
GADD.H.16.N Group add halve signed doublets nearest GADD.H.16.Z Group add halve signed doublets zero GADD.H.32.C Group add halve signed quadlets ceiling GADD.H.32.F Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.N Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets nearest GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.128.Z Group add halve unsigned bytes floor GADD.H.128.F Group add halve unsigned bytes floor GADD.H.128.N Group add halve unsigned bytes nearest GADD.H.128.N Group add halve unsigned doublets ceiling GADD.H.128.N Group add halve unsigned doublets rearest GADD.H.128.N Group add halve unsigned doublets rearest GADD.H.128.C Group add halve unsigned doublets floor GADD.H.128.C Group add halve unsigned quadlets floor GADD.H.128.C Group add halve unsigned quadlets floor GADD.H.128.C Group add halve unsigned quadlets nearest GADD.H.128.C Group add halve unsigned quadlets rearest GADD.H.128.C Group add halve unsigned octlets ceiling GADD.H.128.C Group add halve unsigned octlets ceiling GADD.H.128.C Group add halve unsigned octlets ceiling GADD.H.128.C Group add halve unsigned octlets rearest	GADD.H.16.C	Group add halve signed doublets ceiling
GADD.H.16.Z Group add halve signed doublets zero GADD.H.32.C Group add halve signed quadlets ceiling GADD.H.32.F Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octiets ceiling GADD.H.64.F Group add halve signed octiets floor GADD.H.64.N Group add halve signed octiets nearest GADD.H.64.Z Group add halve signed octiets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.128.Z Group add halve unsigned byter ceiling GADD.H.U.8.C Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.8.N Group add halve unsigned doublets ceiling GADD.H.U.16.C Group add halve unsigned doublets floor GADD.H.U.16.C Group add halve unsigned doublets rearest GADD.H.U.16.N Group add halve unsigned quadlets floor GADD.H.U.16.N Group add halve unsigned quadlets rearest GADD.H.U.16.N Group add halve unsigned quadlets rearest GADD.H.U.16.N Group add halve unsigned quadlets nearest GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.32.N Group add halve unsigned octiets ceiling GADD.H.U.32.N Group add halve unsigned octiets ceiling GADD.H.U.64.P Group add halve unsigned octiets floor GADD.H.U.64.N Group add halve unsigned octiets rearest	G.ADD.H.16.F	Group add halve signed doublets floor
GADD.H.32.C Group add halve signed quadlets ceiling GADD.H.32.F Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octiets ceiling GADD.H.64.F Group add halve signed octiets floor GADD.H.64.N Group add halve signed octiets nearest GADD.H.64.Z Group add halve signed octiets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet nearest GADD.H.0.S.C Group add halve unsigned bytes floor GADD.H.0.S.C Group add halve unsigned bytes floor GADD.H.0.S.N Group add halve unsigned bytes floor GADD.H.0.S.N Group add halve unsigned doublets ceiling GADD.H.0.S.N Group add halve unsigned doublets ceiling GADD.H.0.S.C Group add halve unsigned doublets floor GADD.H.0.S.C Group add halve unsigned doublets floor GADD.H.0.S.C Group add halve unsigned doublets floor GADD.H.0.S.C Group add halve unsigned quadlets ceiling GADD.H.0.S.C Group add halve unsigned quadlets floor GADD.H.0.S.C Group add halve unsigned quadlets floor GADD.H.0.S.C Group add halve unsigned quadlets floor GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets floor GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets floor GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets floor GADD.H.0.S.C Group add halve unsigned octiets rearest GADD.H.0.S.C Group add halve unsigned octiets floor GADD.H.0.S.C GADD.H.0.S.C Group add halve unsigned octiets floor		Group add halve signed doublets nearest
GADD.H.32.F Group add halve signed quadlets floor GADD.H.32.N Group add halve signed quadlets nearest GADD.H.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.N Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets nearest GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.128.Z Group add halve unsigned bytes ceiling GADD.H.128.F Group add halve unsigned bytes floor GADD.H.128.N Group add halve unsigned bytes nearest GADD.H.128.N Group add halve unsigned doublets ceiling GADD.H.128.N Group add halve unsigned doublets floor GADD.H.128.N Group add halve unsigned doublets floor GADD.H.128.C Group add halve unsigned quadlets searest GADD.H.128.C Group add halve unsigned quadlets floor GADD.H.128.N Group add halve unsigned quadlets rearest GADD.H.128.N Group add halve unsigned quadlets floor GADD.H.128.N Group add halve unsigned quadlets rearest GADD.H.128.N Group add halve unsigned octlets rearest GADD.H.128.N Group add halve unsigned octlets ceiling GADD.H.128.N Group add halve unsigned octlets rearest GADD.H.128.N Group add halve unsigned octlets rearest GADD.H.128.C Group add halve unsigned octlets rearest GADD.H.128.C Group add halve unsigned octlets rearest GADD.H.128.C Group add halve unsigned octlets nearest GADD.H.128.C Group add halve unsigned hexlet ceiling		
GADD.H.32.N Group add halve signed quadlets nearest GADD.F.32.Z Group add halve signed quadlets zero GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.N Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.8.N Group add halve unsigned doublets ceiling GADD.H.U.16.C Group add halve unsigned doublets selling GADD.H.U.16.N Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned quadlets ceiling GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.R Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.32.N Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.R Group add halve unsigned octlets rearest GADD.H.U.64.N Group add halve unsigned octlets rearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling		
GADD.H.32.Z Group add have signed quadlets zero GADD.H.64.C Group add have signed octiets ceiling GADD.H.64.F Group add have signed octiets floor GADD.H.64.N Group add have signed octiets nearest GADD.H.64.Z Group add have signed octiets zero GADD.H.128.C Group add have signed hexlet ceiling GADD.H.128.F Group add have signed hexlet floor GADD.H.128.N Group add have signed hexlet nearest GADD.H.128.Z Group add have signed hexlet zero GADD.H.U.8.C Group add have unsigned bytes floor GADD.H.U.8.F Group add have unsigned bytes floor GADD.H.U.8.N Group add have unsigned bytes nearest GADD.H.U.16.C Group add have unsigned doublets ceiling GADD.H.U.16.F Group add have unsigned doublets floor GADD.H.U.16.N Group add have unsigned quadlets ceiling GADD.H.U.13.Z Group add have unsigned quadlets ceiling GADD.H.U.3.Z Group add have unsigned quadlets ceiling GADD.H.U.3.Z Group add have unsigned quadlets floor GADD.H.U.3.Z Group add have unsigned octlets ceiling GADD.H.U.3.Z Group add have unsigned octlets ceiling GADD.H.U.3.Z Group add have unsigned octlets ceiling GADD.H.U.64.C Group add have unsigned octlets rearest GADD.H.U.64.R Group add have unsigned octlets rearest GADD.H.U.64.N Group add have unsigned hexlet ceiling	GADD.H.32.F	Group add halve signed quadlets floor
GADD.H.64.C Group add halve signed octlets ceiling GADD.H.64.F Group add halve signed octlets floor GADD.H.64.N Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned quadlets ceiling GADD.H.U.32.C Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned octlets floor GADD.H.U.32.N Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets floor GADD.H.U.64.C Group add halve unsigned octlets rearest GADD.H.U.64.N Group add halve unsigned octlets rearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C / Group add halve unsigned hexlet ceiling		Group add halve signed quadlets nearest
GADD.H.64.F Group add halve signed octlets floor GADD.H.64.N Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.N Group add halve signed hexlet zero GADD.H.128.Z Group add halve unsigned bytes ceiling GADD.H.U.8.C Group add halve unsigned bytes floor GADD.H.U.8.F Group add halve unsigned bytes nearest GADD.H.U.8.N Group add halve unsigned doublets ceiling GADD.H.U.16.C Group add halve unsigned doublets floor GADD.H.U.16.F Group add halve unsigned doublets mearest GADD.H.U.16.N Group add halve unsigned quadlets ceiling GADD.H.U.32.C Group add halve unsigned quadlets floor GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.R Group add halve unsigned octlets rearest GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet ceiling		
GADD.H.64.N Group add halve signed octlets nearest GADD.H.64.Z Group add halve signed octlets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.N Group add halve signed hexlet zero GADD.H.128.Z Group add halve unsigned bytes ceiling GADD.H.U.8.C Group add halve unsigned bytes floor GADD.H.U.8.F Group add halve unsigned bytes nearest GADD.H.U.8.N Group add halve unsigned doublets ceiling GADD.H.U.16.C Group add halve unsigned doublets floor GADD.H.U.16.F Group add halve unsigned doublets mearest GADD.H.U.16.N Group add halve unsigned quadlets ceiling GADD.H.U.32.C Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.32.N Group add halve unsigned octlets rearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet ceiling		
GADD.H.04.Z Group add halve signed octlets zero GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.08.C Group add halve unsigned bytes floor GADD.H.08.F Group add halve unsigned bytes floor GADD.H.08.N Group add halve unsigned doublets ceiling GADD.H.016.C Group add halve unsigned doublets ceiling GADD.H.016.F Group add halve unsigned doublets floor GADD.H.016.N Group add halve unsigned doublets mearest GADD.H.0132.C Group add halve unsigned quadlets ceiling GADD.H.0132.N Group add halve unsigned quadlets floor GADD.H.0132.N Group add halve unsigned quadlets nearest GADD.H.0132.N Group add halve unsigned octlets ceiling GADD.H.0132.N Group add halve unsigned octlets ceiling GADD.H.0132.N Group add halve unsigned octlets floor GADD.H.0132.N Group add halve unsigned octlets nearest GADD.H.0132.C Group add halve unsigned hexlet ceiling GADD.H.0132.C Group add halve unsigned hexlet ceiling		
GADD.H.128.C Group add halve signed hexlet ceiling GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets mearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.32.N Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet ceiling		
GADD.H.128.F Group add halve signed hexlet floor GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets mearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.32.N Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets floor GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet ceiling		Group add halve signed octlets zero
GADD.H.128.N Group add halve signed hexlet nearest GADD.H.128.Z Group add halve signed hexlet zero GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets mearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.C Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C / Group add halve unsigned hexlet ceiling		
GADD.H.U.8.C Group add halve signed hexlet zero GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets pearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet ceiling		Group add halve signed hexlet floor
GADD.H.U.8.C Group add halve unsigned bytes ceiling GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets mearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C / Group add halve unsigned hexlet floor		
GADD.H.U.8.F Group add halve unsigned bytes floor GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets pearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet floor	GADD.H.128.Z	Group add halve signed hexlet zero
GADD.H.U.8.N Group add halve unsigned bytes nearest GADD.H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets pearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet floor		Group add halve unsigned bytes ceiling
GADD H.U.16.C Group add halve unsigned doublets ceiling GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets bearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor	GADD.H.U.8.F	Group add halve unsigned bytes floor
GADD.H.U.16.F Group add halve unsigned doublets floor GADD.H.U.16.N Group add halve unsigned doublets pearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		Group add halve unsigned bytes nearest
GADD.H.U.16.N Group add halve unsigned doublets bearest GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.C Group add halve unsigned hexlet floor		Group add halve unsigned doublets ceiling
GADD.H.U.32.C Group add halve unsigned doublets bearest GADD.H.U.32.F Group add halve unsigned quadlets ceiling GADD.H.U.32.N Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		
GADD.H.U.32.C Group add halve unsigned quadlets ceiling GADD.H.U.32.F Group add halve unsigned quadlets floor GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		
GADD.H.U.32.N Group add halve unsigned quadlets nearest GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		
GADD.H.U.64.C Group add halve unsigned octlets ceiling GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		Group add halve unsigned quadlets floor
GADD.H.U.64.F Group add halve unsigned octlets floor GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		Group add halve unsigned quadlets nearest
GADD.H.U.64.N Group add halve unsigned octlets nearest GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F Group add halve unsigned hexlet floor		
GADD.H.U.128.C / Group add halve unsigned hexlet ceiling GADD.H.U.128.F / Group add halve unsigned hexlet floor		Group add halve unsigned octlets floor
GADD.H.U.128.F Group add halve unsigned hexlet floor		Group add halve unsigned octlets nearest
		Group add halve unsigned hexlet ceiling
GADD.H.U.128.N Group add halve unsigned heylet nearest		
	G.ADD.H.U.128.N	Group add halve unsigned herlet nearest

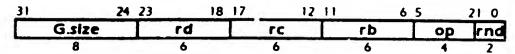
Redundancies

GADD.H.size.rnd rd=rc.rc	⇔ G.COPY rd=rc	
GADD.H.U.size.rnd rd=rc,rc	⇔ G.COPY rd=rc	

Format

G.op.size.rnd rd=rc,rb

rd=gopsizernd(rc,rb)



Description

The contents of registers re and rb are partitioned into groups of operands of the size specified, added, halved, and rounded as specified, yielding a group of results, each of which is the size specified. The results never overflow, so limiting is not required by this operation. The group of results is catenated and placed in register rd.

Z (zero) rounding is not defined for unsigned operations, and a ReservedInstruction exception is raised if attempted. F (floor) rounding will properly round unsigned results downward.

Definition

```
def GroupAddHalvelop,rnd,size,rd,rc,rb)
    c ← RegRead(rc, 128)
    b ← RegRead(rb, 128)
    case op of
         GADDHC, GADDHF, GADDHN, GADDHZ:
              as \leftarrow cs \leftarrow bs \leftarrow 1
         GADDHUC, GADDHUF, GADDHUN, GADDHUZ
              as \leftarrow cs \leftarrow bs \leftarrow 0
              if rnd = Z then
                   raise ReservedInstruction
              endef
    endcase
    h ← size+1
    r - 1
    for i \leftarrow 0 to 128-size by size
         case rnd of
              none, N:
                   s -- Osize 11 -p1
              Z:
                   s ← O<sup>size</sup> 11 P<sub>size</sub>
              F٠
                   s ← Osise+1
              C:
```

s ← O^{size} || || endcase v ← (las & p_{size})||p| + (0||s| a_{size-1+i,1} ← V_{size-1} endfor RegWrite(rd, 128, a) enddef

Exceptions

ReservedInstruction

Group Boolean

These operations take operands from three registers, perform boolean operations on corresponding bits in the operands, and place the concatenated results in the third register.

Operation codes

The second secon		
I G.BOOLEAN	Converted to the conver	
I G'DOOFEAA	Group boolean	i
	taka da karangan da karang	

Equivalencies

GAM	Group three-way and
GMAI	Group add add add bits
GAS.1	Group add add subtract bits
GADD.1	Group add bits
GAND	Group and
GANDN	Group and not
G.COPY	Group copy
G.NAAA	Group three-way nand
G.NAND	Group nand
G.NOOO	Group three-way nor
G.NOR	Group nor
G.NOT	Group not
G.NXXX	Group three-way exclusive-nor
G.000	Group three-way or
G.OR	Group cr
G.ORN	Group or not
G.SAA. I	Group subtract add add bits
G.SAS. 1	Group subtract add subtract bits
G.SET	Group set
G.SET.AND.E.1	Group set and equal zero bits
G.SET.AND.NE.1	Group set and not equal zero bits
G.SET.E. I	Group set equal bits
G.SET.G.1	Group set greater signed bits
G.SET.G.U.1	Group set greater unsigned bits
G.SET.G.Z.1	Group set greater zero signed bits
G.SET.GE. I	Group set greater equal signed bits
G.SET.GE.Z.1	Group set greater equal zero signed bits
G.SET.L.1	Group set less signed bits
G.SET.L.Z.1	Group set less zero signed bits
G.SET.LE. 1	Group set less equal signed bits
G.SET.LE.U.1	Group set less equal unsigned bits
G.SET.LE.Z. 1	Group set less equal zero signed bits
G.SET.NE. 1	Group set not equal bits
G.SET.GE.U.1	Group set greater equal unsigned bits
G.SET.L.U. 1	Group set less unsigned bits

G.SSA 1	Group subtract subtract add bits	
G.SSS. I	Group subtract subtract bits	
G.SUB. I	Group subtract bits	
GXNOR	Group exclusive-nor	
GXOR	Group exclusive-or	
GXXX	Group three-way exclusive-or	
G.ZERO	Group zero	

C 444 40	
GAM rd@rcrb	← G.BOOLEAN rd@rc,rb,0b10000000
GAM I rd@icrb	→ GXXX ref©rc,rb
GAAS.1 rd@rc,rb	→ GXXX rd@rc,rb
GADD.1 rd=rcrb	→ GXOR rd=rc,rb
GAND rd=rc,rb	← G.ROOLEAN rd@rc,rb,0b10001000
GANDN rd=rc,rb	← G.8COLEAN rd@rc,rb,0b01000100
G.BOOLEAN roberty,rc,i	-> 5.BOOLEAN rd@rc,rb,i7i5i6i4i3i1i2i0
G.COPY rd=rc	← G.BOOLEAN rd@rc,rc,0b10001000
G.NAMA rd@rcrb	← G.BOOLEAN rd@rc,rb,0b01111111
G.NAND rd=rc,rb	← G.BOOLEAN rd@rc,rb,0b01110111
G.NOOO rd@rc.rb	← G.BOOLEAN rd@rc,rb,0b00000001
G.NOR rd=rc,rb	← G.BOOLEAN rd@rc,rb,0b00010001
G.NOT rd=rc	← G.BOOLEAN rd@rc,rc,0b00010001
G.NXXX rd@rc,rb	← G.BOOLEAN rd@rc,rb,0b01101001
G.OOO rd@rc,rb	← G.BOOLEAN rd@rc,rb,0b11111110
G.OR rd=rc,rb	← G.BOOLEAN rd@rc,rb,0b11101110
G.ORN rd=rc,rb	← G.BOOLEAN rd@rc,rb,0b11011101
G.SAA.1 rd@rc,rb	-→ GXXX rd@rc,rb
G.SAS.1 rd@rc,rb	→ GXXX rd@rc,rb
G.SET rd	← G.BOOLEAN rd@rd,rd,0b10000001
G.SET.AND.E.1 rd=rb,rc	→ G.NAND rd=rc,rb
G.SET.AND.NE.1 rd=rb,rc	→ GAND rd=rc,rb
G.SET.E.1 rd=rb,rc	→ GXNOR rd=rc,rb
G.SET.G.1 rd=rb,rc	→ GANDN rd=rc,rb
G.SET.G.U.1 rd=rb,rc	→ GANON rd=rb,rc
G.SET.G.Z.1 rd=rc	→ G.ZERO rd
G.SET.GE.1 rd=rb,rc	→ G.ORN rd=rc,rb
G.SE: Gt Z.1 rd=rc	→ G.NOT rd=rc
G.SET.L.i rd=rb,rc	→ GANDN/rd=rb,rc
G.SET.L.Z.1 rd=rc	→ G.COPY rd=rc
G.SET.LE.1 rd=rb,rc	→ G.ORN rd=rb,rc
G.SET.LE.U.1 rd=rb,rc	→ G.ORN rd=rc,rb
G.SET.LE.Z.1 rd=rc	→ G.SET rd
G.SET.NE.1 rd=rb,rc	→ GXOR rd=rc,rb

G.SET.GE.U.1 rd=rb,rc	→	G.ORN rd=rb,rc
G.SET.LU.1 rd=rb,rc	→	GANDN rd=rcrb
G.SSA I rd@rc,rb		GXXX rd@rc,rb
GSSS.1 rd@rc.rb	→	GJXX rd@rc,rb
G.SUB. I rd=rc,rb	\rightarrow	GXOR rd=rc,rb
GUNOR rd=rc,rb	•	G.BOOLEAN rd@rc,rb,0b10011001
GXOR rd=rc.rb	←	G.BOOLEAN rd@rc,rb,0b01100110
GXXX Id@rc.rb		G.BOOLEAN rd@rc,rb,0510010110
G.ZERO rd	←	G.BOOLEAN rd@rd,rd,0b00000000

Selection

operation	function (binary)	function (decimal)
d	11110000	240
C	11001100	204
Ь	10101010	176
d&c&b	10000000	128
(d&=)1b	11101010	234
dicib	11111110	254
₫?c:b	11001010	202
d^c^b	10010110	150
-d^c^b	01101001	105
0	00000000	0

Format

G.BOOLEAN rd@trc,trb,f

rd=gbooleani(rd,rc,rb,f)

31	2524	23	18 17	12 11	6 5	0
G.800	LEAN IN	rd	rc	rb		1
7	1	6	6	6		5

Tue, Aug 17, 1999.

```
if fa=f5 then
                              if fz=f1 then
                                                        if f2 then
                                                                                  rc \leftarrow max[trc,trb]
                                                                                  rb \leftarrow min(trc,trb)
                                                        else
                                                                                  rc \( \tau \) min(trc,trb)
                                                                                  rb \leftarrow max(trc,trb)
                                                        endif
                                                       in ← 0
                                                       il \leftarrow 0 \mid 1 \mid f_6 \mid 1 \mid f_7 \mid 1 \mid f_4 \mid 1 \mid f_3 \mid 1 \mid f_0
                            else
                                                      if f2 then
                                                                                 rc ← trb
                                                                                 rb ← trc
                                                      else
                                                                                 rc ← trc
                                                                                 rb ← trb
                                                      endif
                                                      ih ← 0
                                                      11 \leftarrow 1 \mid 1 \mid f_6 \mid 1 \mid f_7 \mid 1 \mid f_4 \mid 1 \mid f_3 \mid 1 \mid f_0
                            endif
else
                            ih ← 1
                            if fo then
                                                    rc ← trb
                                                      rb ← trc
                                                      il \leftarrow f_1 \cdot | 1 \mid f_2 \mid 1 \mid f_7 \mid 1 \mid f_4 \mid 1 \mid f_3 \mid 1 \mid f_0
                          else
                                                    rc \leftarrow trc
                                                    rb ← trb
                                                    il \leftarrow f_2 + f_3 + f_1 + f_2 + f_3 + f_4 + f_3 + f_4 + f_3 + f_4 + f_3 + f_4 + f_4 + f_5 + f_5 + f_6 +
                          endif
endif
```

Description

Three values are taken from the contents of registers rd, rc and rb. The ih and il fields specify a function of three bits, producing a single bit rest *. The specified function is evaluated for each bit position, and the results are catenated and placed in register rd.

Register rd is both a warree and destination of this instruction.

The function is specified by eight bits, which give the result for each possible value of the three source bits in each bit position:

d	11110000
c	11001100
Ь	10101010
racb)	1 0 1 0 1 0 1 0 f7 f6f5f4f3f2f1f0

A function can be modified by rearranging the bits of the immediate value. The table below shows how rearrangement of immediate value $f_{7..0}$ can reorder the operands d,c,b for the same function.

operation	immediate
fld,c,b)	f7 f6 f5 f4 f3 f2 f1 f0
fic.d.b)	f7 16 f3 f2 f5 f4 f1 f0
f(d,b,c)	f7 f5 f6 f4 f3 f1 f2 f0
f[b,c,d]	f7 f3f5 f1 f6f2 f4f0
f(c,b,d)	f7 f5f3f1f6f4f2f0
fib,d,cj	f7 f3f6f2f5f1f4f0

By using such a rearrangement, an operation of the form: b=f(d,c,b) can be recoded into a legal form: b=f(b,d,c). For example, the function: $b=f(d,c,b)=d\hat{r}c.\hat{\nu}$ cannot be coded, but the equivalent function: $d=c\hat{r}b.d$ can be determined by rearranging the code for $d=f(d,c,b)=d\hat{r}c.b$, which is 11001010, according to the rule for $f(d,c,b) \Rightarrow f(c,b,d)$, to the code 11011000.

Encodina

Some special characteristics of this rearrangement is the basis of the manner in which the eight function specification bits are compressed to seven immediate bits in this instruction. As seen, in the table above, in the general case, a rearrangement of operands from f(d,c,b) to f(d,b,c) (interchanging re and rb) requires interchanging the values of f_0 and f_1 .

Among the 256 possible functions which this instruction can perform, one quarter of them (14 functions) are unchanged by this rearrangement. These functions have the property that f_6 f_5 and $f_2=f_1$. The values of rc and rb^{24} can be freely interchanged, and so are sorted into rising or falling order to indicate the value of f_2 . These functions are encoded by the values of f_7 , f_6 , f_4 , f_3 , and f_0 in the immediate field and f_2 by whether rc>rb, thus using 32 immediate values for 64 functions.

Another quarter of the functions have $f_6=1$ and $f_5=0$. These functions are recoded by interchanging re and rb, f_6 and f_5 , f_2 and f_1 . They then share the same encoding as the

²⁴ Note that rc and rb are the register specifiers, not the register contents.

²⁵ A special case arises when rc=rb, so the sorting of rc and rb cannot convey information. However, as only the values 17, 14, 13, and 10 can ever result in this case, 16, 15, 12, and 11 need not be coded for this case, so no special handling is required.

quarter of the functions where f6=0 and f5=1, and are encoded by the values of f7, f4, f3, f2, f1, and f0 in the immediate field, thus using 64 immediate values for 128 functions.

The remaining quarter of the functions have $f_6=f_5$ and $f_2\ne f_1$. The half of these in which $f_2=1$ and $f_1=0$ are recoded by interchanging re and rb, f_6 and f_5 , f_2 and f_1 . They then share the same encoding as the eighth of the functions where $f_2=0$ and $f_1=1$, and are encoded by the values of f_7 , f_6 , f_4 , f_3 , and f_0 in the immediate field, thus using 32 immediate values for 64 functions.

The function encoding is summarized by the table:

f7	f6	f ₅	14	f3	fz	fı	fo trottb	ih	ils	il4	il3	il2	ilı	ilo	rc	rb
		f6				f ₂	fz	0	0	f6	f7	f4	<i>f</i> ₃	fo	trc.	trb
		f6				f ₂	-f ₂	0	0	f6	17	f4	f 3	fo	trb	trc
		f 6			0	1		0	1	f 6	17	f4	f3	fo	trc,	trb
		fö			1	0		0	1	f ₆	17	f4	f ₃	fo	trb .	trc
	0	1						1	f ₂	fı	f7	f4	f3	fo	trc	trb
	1	0													trb	trc

The function decoding is summarized by the table:

ih	ils	ila	ilz	#2	il;	ilo	rorb	7	f ₆	fs	14	f ₃	12	fi	fo
0	0						0						ō		
0	0						1	il ₃	ila	ila	ilz	ij,	1	1	ib
0	1							#3	ila	il4	ilz	Ħ,	0	1	ilo
1	_							il ₃	0	1	il ₂	ü,	ils	ila	ilo

Definition

```
def GroupBoolean (ih,rd,rc,ris,if)
                                      d ← RegRezd(rd, 128)
                                    c \leftarrow RegRead(rc, 128)
                                    b ← RegRead(rb. 128)
                                    if ih=0 then
                                                                   if 115=0 then
                                                                                                   I \leftarrow H_3 + H_4 + H_4 + H_4 + H_2 + H_4 + H_4 + H_5 + H_6 + 
                                                                   else
                                                                                                   f ← il3 | 1 il4 | 1 il4 | 1 il2 | 1 il1 | 1 | 0 | 1 | 1 | 1 il0
                                                                   endif
                                 else
                                                                 for i/\leftarrow 0 to 127 by size
                                                                si ← [MIRIIN]
                                 endfor
                                 RegWritefrd, 128, at
enddef
```

Exceptions

-

- 147 -

MicroUnity

Group Compare

These operations perform calculations on partitions of bits in two general register values, and generate a fixed-point arithmetic exception if the condition specified is met.

Operation codes

C. C	
G.COM.AND.E.8	Group compare and equal zero bytes
G.COMAND.E.16	Group compare and equal zero doublets
G.COM.AND.E.32	Group compare and equal zero quadlets
G.COMAND.E.64	Group compare and equal zero octions
G.COM.AND.E.128	Group compare and equal zero hexiet
G.COM.AND.NE.B	Group compare and not equal zero bytes
G.COM.AND.NE.16	Group compare and not equal zero doublets
G.COM,AND.NE.32	Group compare and not equal zero quadlets
G.COMAND.NE.64	Group compare and not equal zero octions
G.COMAND.NE.128	Group compare and not equal zero hexlet
G.COM.E.8	Group compare equal bytes
G.COM.E.16	Group compare equal doublets
G.COM.E.32	Group compare equal quadlets
G.COM.E.64	Group compare squal octlets
G.COM.E.128	Group compare equal hexiet
G.COM.GE.8	Group compare greater equal signed bytes
G.COM.GE.16	Group compare greater equal signed doublets
G.COM.GE.32	Group compare greater equal signed quadlets
G.COM.GE.64	Group compare greater equal signed octlets
G.COM.GE.128	Group compare greater equal signed hexlet
G.COM.GE.U.8	Group compare greater equal unsigned bytes
G.COM.GE.U.16	Group compare greater equal unsigned doublets
G.COM.GE.U.32	Group compare greater equal unsigned quadlets
G.COM.GE.U.64	Group compare greater equal cinsigned octies
G.COM.GE.U.128	Group compare greater equal unsigned hexlet
G.COM.L.8	Group compare signed less bytes
G.COM.L.16	Group compare signed less doublets
G.COM.L.32	Group compare signed less quadicis
G COM.L.64	Group compare signed less octlets
G.COM.L.128	Group compare signed less hexlet
G.COM.LU.8	Group compare less unsigned bytes
G.COM.L.U.16	Group compare less unsigned doublets
G.COM.LU.32	Group compare less unsigned quadlets
G.COM.L.U.64	Group compare less unsigned octlets
G.COM.LU.128	Group compare less unsigned hexlet
G.COM.NE.8	Group compare not equal bytes
G.COM.NE.16	Group compare not equal doublets
G.COM.NE.32	Group compare not equal quadlets
G.COM.NE.64	Group compare not equal octies

Instruction Set Group Compare

G.CCA!.NE 128

Group compare not equal hexlet

Equivalencies

G.COM.E.Z.B	Grain compare up al use signed base
G.COM.E.Z.16	Group compare equal zero signed bytes
G.COM.E.Z.32	Group compare equal zero signed doublets
G.COM.E.Z.64	Group compare equal zero signed quadlets
	Group compare equal zero signed octless
G.COM.E.Z.128	Group compare equal zero signed hexlet
G.COM.G.8	Group compare signed greater bytes
G.COM.G.16	Group compare signed greater doublets
G.COM.G.32	Group compare signed greater quadlets
G.COM.G.64	Group compare signed greater citlets
G.COM.G.128	Group compare signed greater hexlet
G.COM.G.U.8	Group compare greater unsigned bytes
G.COM.G.U.16	Group compare greater unsigned doublets
G.COM.G.U.32	Group compare greater unsigned quadlets
G.COM.G.U.64	Group compare greater unsigned octies
G.CCM.G.U.128	Group compare greater unsigned hedet
G.COM.GZ.8	Group compare greater zero signed bytes
G.COM.G.Z.16	Group compare greater zero signed doublets
GCOM.G.Z.32	Group compare greater zero signed quadlets
G.COM.G.Z.64	Group compare greater zero signed ocuets
G.COM.G.Z.128	Group compare greater zero signed hexlet
G.COM.GEZ.8	Group compare greater equal zero signed bytes
G.COM.GEZ.16	Group compare greater equal zero signed doublets
G.COM GE 2.32	Group compare greater equal zero signed quadlets
G.COM.GE.Z.64	Group compare greater equal zero siged octlets
G.COM.GE Z. 128	Group compare greater equal zero signed hexlet
G.COM.L.Z.8	Group compare less zero signed bytes
G.COM.L.Z.16	Group compare less zero signed doublets
G.COM.L.Z.32	Group compare less zero signed quadlets
G.COM.L.Z.64	Group compare less zero signed octlets
G.COM.L.Z.128	Group compare less zero signed hexlet
G.COM.LE.8	Group compare less equal signed bytes
G.COM.LE. 16	Group compare less equal signed doublets
G.COM.LE.32	Group compare less equit signed quadlets
G.COMLE.64	Group compare less equal signed octless
G.COM.LE.128	Group compare less equel signed hexiet
G.COM.LE.U.8	Group compare less equal unsigned bytes
G.COM.LE.U.16	Group compare less equal unsigned doublets
G.COM.LE.U.32	Group compare less equal unsigned quadlets
G.COM.LE.U.64	Group compare less equal unsigned octlets
G.COM.LE.U.128	Group compare less equal unsigned hexlet
G.COM.LE.Z.8	Group compare less equal zero signed bytes
G.COM.LE.Z.16	Group compare less equal zero signed doublets
G.CCM.LE.Z.32	Group compare less equal zero signed quadlets
G.COM.LE.Z.64	Group compare less equal zero signed octlets
	1

G.COMLEZ.128	Group compare less equal zero signed hexlet
G.COMNEZ.8	Group compare not equal zero signed bytes
G.COM.NE.Z.16	Group compare not equal zero signed doublets
G.COM.NE.Z.32	Group compare not equal zero signed quadlets
G.COMNEZ.64	Group compare not equal zero signed octlets
G.COM.NE.Z.128	Group compare not equal zero signed hexlet
G.FIX	Group fixed point arithmetic exception
G.NOP	Group no operation

G.COM.E.Z.size rc	←	G.COMAND.E.size rc,rc
G.COM.G.size rd,rc	→	G.COM.L.size rc.rd
G.COM.G.U.size rd,rc	→	G.COM.L.U.size rc,rd
G.COM.G.Z.size rc	=	G.COM.L.U.size rc,rc
G.COM.GEZ.stze rc	=	G.COM.GE.size rc,rc
G.COM.L.Z.size rc	=	G.COM.L.size rc.rc
G.COMLE.size rd,rc	→	G.COM.GE.size rc.rd
G.COMLE.U.size rd,rc	→	G.COM.GE.U.size rc,rd
G.COM.LE.Z.size rc	(G.COM.GE.U.size rc.rc
G.COM.NE.Z.size rc	(-	G.COMAND.NE.size rc,rc
G.FIX	-	G.COM.E.128 r0,r0
G.NOP	←	G.COM.NE.128 r0,r0

Redundancies

G.COM.E.size rd,rd	⇔ G.FDX	
G.COM.NE.size rd,rd	⇔ G.NOP	

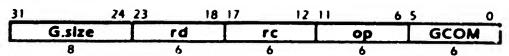
Selection

class	operation	cond	type	size					
boolean	COMAND	E NE			8	16	32	64	128
arithmetic	COM	L GE G LE	NONE U		8	16	32	64	128
<u> </u>	COM	L GE G LE E NE	Z		8	16	32	64	128

Eormat

G.COM.op.size rd,rc G.COM.opz.size rcd

gcomopsize(rd,rc)



Description

Two values are taken from the contents of registers rd and rc. The specified condition is calculated on partitions of the ope ands. If the specified condition is true for any partition, a fixed-point arithmetic exception is generated. This instruction generates no general purpose register results...

Definition

enddef

```
def GroupCompare(op,size,rd,rc)
              d ← RegReadird, 128)
              c ← RegRead(rc, 128)
              case op of
                              G.COM.E:
                                             for i \leftarrow 0 to 123-size by size
                                                             منهودا ، ← الأسهودا ، = دسهودا ، أعدو
                                             endfor
                             G COM NE:
                                             for i ← 0 to 128-size by size
                                                            Suspenior (duspenior Chareningsize
                                             endfor
                             G.COMAND.E.
                                            for i \leftarrow 0 to 128-size by size
                                                            givers ! - !Kindsen ! and dindsent ! = Oferse
                             G.COM.AND.NE
                                            for i ← 0 to 128-size by size
                                                            district + (Kinggert; and district # Office
                                            endiar
                            G.COM.L:
                                            for i \leftarrow 0 to 128-size by size
                                                            &+size-1 ; ← (frd = rc) ? (c+size-1,1 < 0) : (d+size-1 ; < c+size-1 ; ) size
                                            endfor
                            G COM GE
                                           for i \leftarrow 0 to 128-size by size
                                                            مهم عندود 1 را الاو = ددا ع الاستعداد ع 1 المستعدد 1 ا ح د المناعد الأواء المناعد الله عدد المناعد الله المناطقة المنا
                                            endlor
                            G COM.L.U
                                           ior i ← 0 to 128-size by size
                                                           Spossen : ← (Ird = rc) ? (Cpossen : > 0) :
                                                                          ((0 11 dosie-1 ) < (U 11 Cheste-1 ))) size
                                           endfor
                           G COM GE U
                                           for i ← 0 to 128-size by size
                                                          ajouze-1 : ← ((rd = rc) ? (c++9ze-1 ... ≤ 0) :
                                                                          10 11 questo-1 9 5 10 11 chate-1 9) ase
                                           enc. x
           endcase
            # la = 01 then
                          raise FixedPointArithmetic
           endif
```

Instruction Set Group Compase

Exceptions

Fined-pour andumetic

Group Compare Floating-point

These operations perform calculations on partitions of bits in two general register values, and generate a floating point antimetic exception if the condition specified it met.

Operation codes

G.COM.E.F. 16	, Group compare inqual floating-point half
G.COM.E.F.16X	Group compare equal floating-point half much
G.COM.E.F.32	Group compare inqual floating-point ungle
G.COM.E.F.32.X	Group compare equal finance, point single exist
G.COM.E F.64	Group compare equal floating point double
G.COM.E.F.64X	Group compare equal floating-point double exist
G.COM.E.F.128	Group compare equity floating worst quad
G.COME.F. 128X	Group compare equal floating-point quad exact
G.COM.GE.F.16	Group compare greater or equal floriting-point half
G.COM.GE.F.16X	Group compare oreaser or equal floating-point half exact
G.COM.GE.F.32	Crosto compare greater or equal floating-point single
C.COM.GE.F.32X	Group compare greater or equal floating-point single exact
G.COM GE.F.64	Group compare greater or equal floating-point double
G.COM.GE.F.64.X	Group compare greater or equal floating-point double exact
G.COM.GE.F.128	Group compare greater or equal floating point quad
G.COM.GE.F. 128.X	Group compare greater or equal finating-puint quad exact
G.CCM.L.F.16	Group compare I'm floating-point half
G.COM.L.F.16X	Group compare instituating point half exact
G.COM.L.F.32	Croup compare less floriting-point sincle
G.COM.L.F.32.X	Group compare less floating point single exact
G.COM.LF.64	Group compare less floating point devale
G.COM.L.F.64.X	Group compare 345 floating point double exact
G.COM.L.F.128	Group compare has floating-point quad
G.COM.L.F.128.X	Group company less hoading-point quad exact
G.COM.LG.F.16	Group compare less or greater floating-point half
G.COM.LGF.16X	Group compare less or greater floating-point half exact
GCOMLGF32	Group compare less or preater flowing-point single
G.COM.LG.F.3?X	Group compare less or greater floating-comit single exact
G COMLGF 64	Group compare les de creater ficultang point double
C.COMLG.F.64X	Group compare less or greater finating-point double exact
G.COM.LG.F.128	Froup compare less or greater fluating-point quad
G.COM.LG.F.128X	Group compare less or greater floating point qu'il mact

Equivalencies

G COM.G.F.16	Group compare greater floating-point half
G.COM.G.F. 16.X	Group compare greater floating-point half exact
G.COM.G.F.32	Group compare greater floating-point single
G.COM.G.F.32X	Group compare greater floating-point single exact
G.COM.G.F.64	Group compare greater floating-point double
G.COM.G.F.64X	Group compare greater floating-point double exact
G.COM.G.F.128	Group compare greater floating-point quad
G.COM.G.F.128X	Group compare greater floating-point quad exact
G.COM.LE.F.16	Group compare less equal floating-point half
G.COM.LE.F. 16X	Group compare less equal floating-point half exact
G.COM.LE.F.32	Group compare less equal floating-point single
G.COM.LE.F.32X	Group compare less equal floating-point single exact
G.COM.LE.F.64	Group compare less equal floating-point double
G.COM.LE.F.64X	Group compare less equal floating-point double exact
G.COM.LE.F.128	Group compare less equal floating-point quad
G.COM.LE.F. 128X	Group compare less equal floating-point quad exact

G.COM,G.F.prec rd,rc	→	G.COM.L.F.prec rc,rd	
G.COM.G.F.prec.X rd,rc	→	G.COM.L.F.prec.X rc,rd	
G.COM.LE.F.prec rd,rc	→	G.COM.GE.F.prec rc,rd	
G.COM.LE.F.prec.X rd,rc	\rightarrow	G.COM.GE.F.prec.X rc,rd	

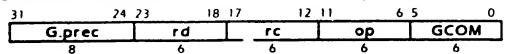
Selection

class	ор	cond	type	prec	round/trap
set	COM	E LG L GE G LE	F	16 32 64 128	NONE X

Format

G.COM.op.prec.round rd,rc

rc=gcomopprecround(rd,rc)



Description

The contents of registers rd and rc are compared using the specified floating-point condition. If the result of the comparison is true for any corresponding pair of elements, a floating point exception is raised. If a rounding option is specified, the operation raises a floating point exception if a floating point invalid operation occurs. If a rounding option is not specified, floating point exceptions are not raised, and are handled according to the default rules of IEEE 754.

Definition

```
def GroupCompareFloatingPoint(op,prec,round,rd,rc) as
     d ← RegRead(rd, 128)
     c ← RegRead(rc, 128)
     for i \leftarrow 0 to 128-prec by prec
          di \leftarrow F[prec, d_{i+prec-1}, i]
          ci - Fiprec, Ci-prec-1.d
           if round#NONE then
                if |di.t = SNAN| or |ci.t = SNAN| then
                     raise FloatingPointAnthmetic
                endif
                case op of
                     G.COM.L.F., G.COM.GE.F:
                          # (di.t = QNAN) or (ci.t = QNAN) then
                                raise FloatingPointArithmetic
                     others: //nothing
                endcase
          endif
          case op of
                G.COM.LF:
                     a ← d172c1
                G.COM.GE.F:
                     ar - dilîkci
               G.COM E.F.
                     al - di=ci
               G.COM LG.F:
                     a - deci
          endcase
          amprec-1 1 ← al
     endfor
     if (a \neq 0) then
          raise FloatingPointArithmetic
     endif
enddef
```

Exceptions

Heating point arithmetic

Group Copy Immediate

This operation copies an immediate value to a general register.

Operation codes

G.COPY.I.16	Group copy immediate doublet
G.COPY.I.32	Group signed copy immediate quadlet
G.COPY.I.64	Group signed copy immediate octlet
G.COPY.I.128	Group signed copy immediate hexlet

Equivalencies

G.COPY.I.8	Group copy immediate byte
G.SET	Group set
G.ZERO	Group zero

G.COPY.1.8 rd= i3 1 i70	←	G.COPY.I.16 rd=(0 1 i70 1 i70)
GSET rd	←	G.COPY.I.128 rd=-1
G.ZERO rd	←	G.COPY.I.128 rd=0

Redundancies

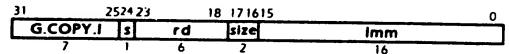
G.COPY.I.size rd=-1	⇔	G.SET rd	7
G.COPY.I.size rd=0	⇔	G.ZERO rd	ᅦ

Format

G.COPY.I.size

rd=i

rd=gcopyisize(i)



 $s \leftarrow i_{16}$ imm $\leftarrow i_{150}$

Description

 Λ 128-bit immediate value is produced from the operation code, the size field and the 16-bit imm field. The result is placed into register ra.

Definition

def GroupCopyImmediate(op,size,rd,imm) as

 $s \leftarrow op_0$

```
case size of 16:

If s then ReservedInstruction endif

a \leftarrow imm + 1 \cdot imm +
```

Exceptions

Reserved Instruction

Group Immediate

These operations take operands from a register and an immediate value, perform operations on partitions of bits in the operands, and place the concatenated results in a second register.

Operation codes

	·
GADD.L16	Group add immediate doublet
GADD.I. 16.0	Group add immediate signed doublet check overfluw
GADD.I.32	Group add immediate quadlet
G.ADD.I.32.O	Group add immediate signed quadlet check overflow
GADD.I.64	Group add immediate octlet
GADD.1.64.O	Group add immediate signed octlet check overflow
GADD.I.128	Group add immediate hexlet
G.ADD.I.128.O	Group add immediate signed hexlet check overflow
G.ADD.I.U.16.O	Group add immediate unsigned doublet check overflow
GADD.I.U.32.O	Group add immediate unsigned quadlet check overflow
GADD.I.U.64.O	Group add immediate unsigned octlet check overflow
GADD.I.U.128.O	Group add immediate unsigned hexlet check overflow
GAND.I.16	Group and immediate doublet
GAND.I.32	Group and immediate quadlet
G.AND.1.64	Group and immediate octlet
G.AND.I.128	Group and immediate hexlet
G.NAND.I.16	Group not and immediate doublet
G.NAND.I.32	Group not and immediate quadlet
G.NAND.I.64	Group not and immediate octlet
G.NAND.I.128	Group not and immediate hexlet
G.NOR.I.16	Group not or immediate doublet
G.NOR.I.32	Group not or immediate quadlet
G.NOR.1.64	Group not or immediate octlet
G.NOR.I.128	Group not or immediate hexlet
G.OR.I.16	Group or immediate doublet
G.OR.J.32	Group or immediate quadlet
G.OR.I.64	Group or immediate octlet
G.OR.I.128	Group or immediate hexlet
GXORI.16	Group exclusive-or immediate doublet
GXOR.I.32	Group exclusive-or immediate quadlet
GXOR.I.64	Group exclusive-or immediate octlet
G.XOR.I.128	Group exclusive-or immediate hexlet

Equivalencies

GANDN.I.16	Group and not immediate doublet
GANDN.1.32	Group and not immediate quadlet
G.ANDN.I.64	Group and not immediate octlet
GANDN.I.128	Group and not immediate hexlet
G.COPY	Group copy
G.NOT	Group not
G.ORN.I.16	Group or not immediate doublet
G.ORN.1.32	Group or not immediate quadlet
G.ORN.I.64	Group or not immediate octlet
G.ORN.I.128	Group or not immediate hexlet
GXNOR.I.16	Group exclusive-nor immediate doublet
GXNOR.1.32	Group exclusive-nor immediate quadlet
GXNOR.1.64	Group exclusive-rior immediate octlet
GXNOR.1.128	Group exclusive-nor immediate hexlet

GANDN.I.size rd=rc 71	→	GAND.I.size rd=rc,-imm	
G.COPY rd=rc	←	G.OR.I.128 rd=rc,0	
G.NOT rd=rc	←	G.NOR.I.128 rd=rc.0	
G.ORN.I.size rd=rc.imm	→	G.OR.I.size rd=rc,~imm	
GXNOR.I.size rd=rc.imm	→	GXOR.I.size rd=rc,-imm	

Redundancies

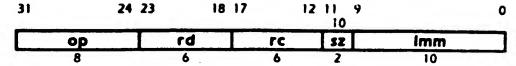
Contraction

GADD.I.size rd=rc,0	⇔	G.COPY rd=rc
GADD.I.size.O rd=rc,0	⇔	G.COPY rd=rc
GADD.I.U.size.O rd=rc,0	⇔	G.COPY rd=rc
GAND.I.size rd=rc,0	⇔	G.ZERO rd
GAND.I.size rd=rc,-1	⇔	G.COPY rd=rc
G.NAND.I.size rd=rc,0	⇔	G.SET rd
G.NAND.I.size rd=rc,-1	⇔	G.NOT rd=rc
G.OR.I.size rd=rc,-1	⇔	G.SET rd
G.NOR.I.size rd=rc,-1	⇔	G.ZERO rd
GXOR.I.size rd=rc,0	⇔	G.COPY rd=rc
GXOR.I.size rd=rc,-1	€⇒	G.NOT rd=rc

Eormat

op.size rd=rc,imm

rd=opsize(rc,imm)



 $sz \leftarrow log(size)-4$

Description

The contents of register re is fetched, and a 128-bit immediate value is produced from the operation code, the size field and the 10-bit imm field. The specified operation is performed on these operands. The result is placed into register ra.

Definition

- : · . : : : ·

```
def GroupImmediate(op,size,rd.rc,imm) as
     c ← RegRead(rc, 128)
     s ← immg
     case size of
           16:
                 i16 ← s<sup>7</sup> | | imm:
                 b ← i16 11 i16
                 b \leftarrow s^{22} + 11 \text{ imm} + 11 \text{ s}^{22} + 11 \text{ imm} + 11 \text{ s}^{22} + 11 \text{ imm}
           64:
                 b \leftarrow s^{54} + l + imm + l + s^{54} + l + imm
           128:
                 b ← s118 | 1 mm
     endcase
     case op of
           GAND.I:
                 a ← c and b
           G.ORJ:
                 a \leftarrow c \text{ or } b
           G NAND.I:
                 a ← c nand b
           G.NOR.I:
                 a ← c nor b
           G.XOR.I:
                a c xor b
           GADD.I:
                 for i \leftarrow 0 to 128-size by size
                       ansize-1... ← Cinsize-1... + binsize-1...
                 e. IL Yor
           GADD.I.U:
                for i \leftarrow 0 to 128-size by size
                      1 ← (Ciosize-1 | 1 Ciosize-1 | 1 biosize-1 | 1 biosize-1 | 1
```

Fixed-point anthmetic

```
if t<sub>size</sub> ≠ t<sub>size-1</sub> then
raise FixedPointArithmetic
endiff

āli-size-1...i ← t<sub>size-1...0</sub>

endfor

GADD.I.U.O:

for i ← 0 to 128-size by size

t ← [0] | 1 t<sub>i-size-1...il</sub> + [0] | 1 b<sub>i-size-1...il</sub>

if t<sub>size</sub> ≠ 0 then
raise FixedPointArithmetic
endiff
āli-size-1...i ← t<sub>size-1...0</sub>

endfor
endcase
RegWrite[rd, 128, a]
enddef

Exceptions
```

Group Immediate Reversed

These operations take operands from a register and an immediate value, perform operations on partitions of bits in the operands, and place the concatenated results in a second register.

Operation codes

G.SET.AND.E.I.16	Group set and equal zero immediate doublets
G.SET.AND.E.I.32	Group set and equal zero immediate quadlets
G.SET.AND.E.I.64	Group set and equal zero immediate octlets
G.SET.AND.E.I.128	Group set and equal zero immediate hexlet
G.SET.AND.NE.I.16	Group set and not equal zero immediate doublets
G.SET.AND.NE.1.32	Group set and not equal zero immediate quadlets
G.SET.AND.NE.I.64	Group set and not equal zero immediate octlets
G.SET.AND.NE.I.128	Group set and not equal zero immediate hexlet
G.SET.E.I.16	Group set equal immediate doublets
G.SET.E.I.32	Group set equal immediate quadlets
G.SET.E.I.64	Group set equal immediate octlets
G.SET.E.I. 128	Group set equal immediate hexlet
G.SET.GE.I.16	Group set greater equal immediate signed doublets
G.SET.GE.I.32	Group set greater equal immediate signed quadlets
G.SET.GE.I.64	Group set greater equal immediate signed octlets
G.SET.GE.I.128	Group set greater equal immediate signed hexlet
GSET.GE.I.U.16	Group set greater equal immediate unsigned doublets
G.SET.GE.I.U.32	Group set greater equal immediate unsigned quadlets
G.SET.GE.I.U.64	Group set greater equal immediate unsigned octlets
G.SET.GE.I.U. 128	Group set greater equal immediate unsigned hexlet
G.SET.L.I.16	Group set signed less immediate doublets
G.SET.L.1.32	Group set signed less immediate quadlets
GSET.L1.64	Group set signed less immediate octlets
GSETLI.128	Group set signed less immediate hexlet
G.SET.L.I.U.16	Group set less immediate signed doublets
G.SET.L.I.U.32	Group set less immediate signed quadlets
G.SET.L.I.U.64	Group set less immediate signed octlets
G.SET.L.I.U.128	Group set less immediate signed hexlet
G.SET.NE.I.16	Group set not equal immediate doublets
G.SET.NE.I.32	Group set not equal immediate quadlets
G.SET.NE.I.64	Group set not equal immediate octlets
G.SET.NE.I.128	Group set not equal immediate hexlet
G.SUB.I.16	Group subtract immediate doublet
G.SUB.I.16.O	Group subtract immediate signed doublet check overflow
G.SUB.I.32	Group subtract immediate quadlet
G.SUB.1.32.O	Group subtract immediate signed quadlet check overflow
G.SUB.1.64	Group subtract immediate octlet
G.SUB.1.64.O	Group subtract immediate signed octlet check overflow
G.SUB.I.128	Group subtract immediate hexlet

G.SUB.I.128.O	Group subtract immediate signed hextet check overflow
G.SUB.I.U. 16.O	Group subtract immediate unsigned doublet check overflow
G.SUB.I.U.32.O	Group subtract immediate unsigned quadlet check overflow
G.SUB.I.U.64.O	Group subtract immediate unsigned octlet check overflow
G.SUB.I.U.128.O	Group subtract immediate unsigned hexlet check overflow

Equivalencies

G.NEG.16	Group negate dc.:blet
G.NEG. 16.0	Group negate signed doublet check overflow
G.NEG.32	Group negate quadlet
G.NEG.32.0	Group negate signed quadlet check overflow
G.NEG.64	Group negate octlet
G.NEG.64.0	Group negate signed octlet check overflow
G.NEG.128	Group negate heidet
G.NEG. 128.0	Group negate signed hexlet check overflow
G.SET.LE.I.16	Group set less equal immediate signed doublets
G.SET.LE.1.32	Group set less equal immediate signed quadlets
G.SET.LE.I.64	Group set less equal immediate signed octiets
G.SET.LE.I.128	Group set less equal immediate signed healet
G.SET.LE.I.U.16	Group set less equal immediate unsigned doublets
G.SET.LE.I.U.32	Group set less equal immediate unsigned quadlets
G.SET.LE.I.U.64	Group set less equal immediate unsigned octlets
G.SET.LE.i.U.128	Group set less equal immediate unsigned hextet
G.SE1.G.I.16	Group set immediate signed greater doublets
G.SET.G.1.32	Group set immediate signed greater quadlets
G.SET.G.1.64	Group set immediate signed greater octlets
G.SET.G.I.128	Group set immediate signed greater hexlet
G.SET.G.I.U.16	Group set greater immediate unsigned doublets
G.SET.G.I.U.32	Group set greater immediate unsigned quadlets
G.SET.G.I.U.64	Group set greater immediate unsigned octiets
G.SET.G.I.U.128	Group set greater immediate unsigned hexlet

G.NEG.size ra=rc	→ ASUB.I.size rd=0,rc	
G.NEG.size.O rd=rc	→ ASUB.I.size.O rd=0,rc	
G.SET.G.I.size rd=imm,rc	→ G.SET.GE.I.size rd=imm+1,rc	
G.SET.G.I.U.size rd=imm.rc	→ G.SET.GE.I.U.size rd=irnm+1,rc	
G.SET.LE.I.size rd=imm,rc	→ G.SET.L.I.size rd=imm-1,rc	
G.SET.LE.I.U.size rd=imm,rc	→ G.SET.L.I.U.size rd=irrim-1,rc	

Redundancies

G.SET.AND.E.I.size rd=rc,0	₩	G.SET.size rd	
G.SET.AND.NE.I.size rd=rc.0	⇔	G.ZERO rd	
G.SET.AND.E.I.size rd=rc,-1	⇔	G.SET.E.Z.size rd=rc	
G.SET.AND.NE.I.size rd=rc,-1	⇔	G.SET.NE.Z.size rd-rc	
G.SET.E.I.size rd=rc,0	⇔	G.SET.E.Z.size rd=rc	
G.SET.GE.Lsize rd=rc,0	⇔	G.SET.GE.Z.size rd=rc	
GSET.L.I.size rd=rc,0	⇔	G.SET.L.Z.size rd=rc	
G.SET.NE.I.size rd=rc,0	⇔	G.SET.NE.Z.size rd=rc	
GSET.GE.LU.size rd=rc.0	⇔	G.SET.GE.U.Z.size rd=rc	
GSET.L.I.U.size rd=rc.0	⇔	G.SET.L.U.Z.size rd=rc	

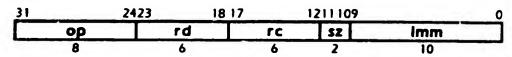
Selection

class	operation	cond	form	operand	size	check
arithmetic	SUB		1		16 32 64 12	28
		1		NONE L	16 32 64 12	28 0
boolean	SET.AND SET	E NE	1		16 32 64 12	28
	SET	L GE G LE	1	NONE L	16 32 64 12	28

Format

op.size rd=imm.rc

rd=opsize(imm,rc)



SZ +- log(size)-4

Description .

The contents of register rc is fetched, and a 128-bit immediate value is produced from the operation code, the size field and the 10-bit imm field. The specified operation is performed on these operands. The result is placed into register rd.

Definition

def GroupImmediateReversed(op,size,ra,imm) as

c - RegRead(rc, 128)

s ← imm9

case size of

16:

```
116 ← s<sup>7</sup> 11. imm
                                   b ← i16 11 i16
                  32:
                                   b \leftarrow s^{22} + 11 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 \text{ } s^{22} + 1 \text{ imm } 11 
                 64:
                                   b \leftarrow s^{54} 11 imm 11 s^{54} 11 imm
                  128:
                                  b ← s118 !! imm
endcase
case op of
                G.SUBJ:
                                  for i \leftarrow 0 to 128-size by size
                                                    Bi+size-1...i ← Di+size-1...i - Ci+size-1..i
                G.SUBJ.O:
                                  for i \leftarrow 0 to 128-size by size
                                             t - (Opgze-1 !! Opgize-1.il - (Cpgize-1 !! Cpgize-1.il
                                                   if figize # taize-1 then
                                                                    raise FixedPointAnthmetic
                                                   endil
                                                   dissize-1..i ← lsize-1..0
                                 endfor
               G.SUBJ.U.O.
                                 for i \leftarrow 0 to 128-size by size
                                                  t \leftarrow [0^1 \ 11 \ D_{\text{trigge-1...}}] - [0^1 \ 11 \ C_{\text{trigge-1...}}]
                                                  if Rize # 0 then
                                                                   raise FixedPointArithmetic
                                                  Ausize-1..i ← Caze-1..0
                                endfor
               G.SET.E.I:
                                for i \leftarrow 0 to 128-size by size
                                                 dispelui ← (Dispelui = Cisseluisize
                               endfor
              G.SET.NE.I:
                               for i \leftarrow 0 to 128-size by size
                                                 Sugge-1... ← Dissize-1... * Cusize-1... Size
                               endfor
              G.SET.AND.E.I:
                               for i \leftarrow 0 to 128-size by size
                                                and circle | [iphoise-1 i and circle-1 ii = Ofsize
                               endfor
             G.SET.AND.NE.I:
                              for i \leftarrow 0 to 128-size by size
                                                and Cingige-1, i ≠ Ofsize
                              endfor
             G.SET.L.I:
                              for i \leftarrow 0 to 128-size by size
                                               angize-1... ← (Dinsize-1... < Consize-1...) Size
                              endfor
            G.SET.GE.I:
                             for i \leftarrow 0 to 128-size by size
```

```
al-size-1_i \( \text{[Division 1_i]} \geq \text{Civisize-1_ii} \for i \)

for i \( \text{O} \) to 128-size by size

al-size-1_i \( \text{[O 11 Divisize-1_ii} < \text{[O 11 Civisize-1_iii} \)

endfor

G.SET.GE.I.U:

for i \( \text{O 10 128-size by size} \)

al-size-1_i \( \text{[O 11 Divisize-1_iii} \geq \text{[O 11 Civisize-1_iii} \)

endfor

endcase

RegWrite[rd, 128, a]
enddef
```

Exceptions

hixed-point anthmetic

Group Inplace

These operations take operands from three registers, perform operations on partitions of bits in the operands, and place the concatenated results in the third register.

Operation codes

GAMA8	Group add add add bytes
GAM 16	Group add add doublets
GAM32	Group add add guadlets
GAM64	Group add add octlets
GAM 128	Group add add add hexlet
GASA8	Group add subtract add bytes
GASA 16	Group add subtract add doublets
GASA 32	Group add subtract add quadlets
GASA 64	Group add subtract add octlets
GASA 128	Group add subtract add hexlet

Equivalencies

GANS8	Group add add subtract bytes
GMS.16	Group add add subtract doublets
GANS.32	Group add add subtract quadlets
GMS.64	Group add add subtract octlets
GAAS. 128	Group add add subtract hexlet

		والمتحادثات الأماء والمتحارب والمتحا
CAAC		GASAsize rd@rb,rc
GANS.size rd@rc.rb	>	CANNE INGINIC

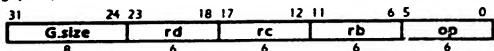
Redundancies

G.M.size rd@rc.rc	==	GSHLIADD.size rd=rd,rc,1
GASA size rd@rc.rc	0	G.NOP

Econnat

G.op.sizerd@rc,rb

rd=gopsize(rd,rc,rb)



Description

The contents of registers rd, rc and rb are fetched. The specified operation is performed on these operands. The result is placed into register rd.

Register rd is both a source and destination of this instruction.

Definition

none

The wind with the same of the com-

```
del Groupinpiace(op.size,rd.rc.rb) as

d ← RegRead(rd. 128)

c ← RegRead(rd. 128)

b ← RegRead(rb. 128)

for i ← 0 to 128-size by size

case op of

G.AAA:

alosize-1_i ← c diosize-1_i + Ciosize-1_i + biosize-1_i

G.ASA:

alosize-1_i ← + diosize-1_i - Ciosize-1_i + biosize-1_i

endcase

endfor

RegWrite(rd. 128. a)

enddef

Exceptions
```

MicroUnity

Instruction Set Group Reversed

Group Reversed

These operations take two values from registers, perform operations on partitions of bits in the operands, and place the concatenated results in a register.

Operation codes

G.SET.AND.E.8	Group set and equal zero bytes
G.SET.AND.E.16	Group set and equal zero doublets
G.SET.AND.E.32	Group set and equal zero quadlets
G.SET.AND.E.64	Group set and equal zero octiets
G.SET.AND.E.128	Group set and equal zero hexlet
G.SET.AND.NE.8	Group set and not equal zero bytes
G.SET.AND.NE.16	Group set and not equal zero doublets
G.SET.AND.NE.32	Group set and not equal zero quadlets
G.SET.AND.NE.64	Group set and not equal zero octiets
G.SET.AND.NE.128	Group set and not equal zero hexlet
G.SET.E.8	Group set equal bytes
G.SET.E.16	Group set equal doublets
G.SET.E.32	Group set equal quadlets
G.SET.E.64	Group set equal octlets
G.SET.E.128	Group set equal hextet
G.SET.GE.8	Group set greater equal signed bytes
G.SET.GE.16	Group set greater equal signed doublets
G.SET.GE.32	Group set greater equal signed quadlets
G.SET.GE.64	Group set greater equal signed octlets
G.SET.GE.128	Group set greater equal signed hexlet
G.SET.GE.U.8	Group set greater equal unsigned bytes
G.SET.GE.U.16	Group set greater equal unsigned doublets
G.SET.GE.U.32	Group set greater equal unsigned quadiets
G.SET.GE.U.64	Group set greater equal unsigned octiets
G.SET.GE.U.128	Group set greater equal unsigned hexlet
G.SET.L.8	Group set signed less bytes
G.SET.L.16	Group set signed less doublets
G.SET.L32	Group set signed less quadlets
G.SET.L.64	Group set signed less octlets
G.SET.L 128	Group set signed less hexlet
G.SET.LU.8	Group set less unsigned bytes
G.SET.L.U.16	Group set less unsigned doublets
G.SET.LU.32	Group set less unsigned quacilets
G.SET.LU.64	Group set less unsigned octles /
G.SET.L.U.128	Group set less unsigned hexlet
G.SET.NE.8	Group set not equal bytes
G.SET.NE.16	Group set not equal doublets
G.SET.NE.32	Group set not equal quadlets
G.SET.NE.64	Group set not equal octlets

G.SET.NE. 128	Group set not equal hextet
G.SUB.8	Group subtract bytes
G.SUB.B.O	Group subtract signed bytes check overflow
GSUB 16	Group subtract doublets
GSUB 16.0	Group subtract signed doublets check overflow
GSUB.32	Group subtract quadlets
GSUB32.0	Group subtract signed quadlets check overflow
G.SUB.64	Group subtract octlets
GSUB.64.0	Group subtract signed octiets check overflow
GSUB.128	Group subtract heidet
G.SUB. 128.O	Group subtract signed hextet check overflow
GSUBL8	Group subtract limit signed bytes
GSUBT 19	Group subtract limit signed doublets
GSUBL 32	Group subtract limit signed quadlets
GSUBL 64	Group subtract limit signed octlets
GSUBL 128	Group subtract limit signed hexlet
GSUBLU.8	Group subtract limit unsigned bytes
GSUBLU.16	Group subtract limit unsigned doublets
GSUBLU32	Group subtract limit unsigned quadlets
G.SUBLU.64	Group subtract limit unsigned octlets
GSUBLU.128	Group subtract limit unsigned heidet
G.SUB.U.8.O	Group subtract unsigned bytes check overflow
G.SUB.U.16.0	Group subtract unsigned doublets check overflow
G.\$UB.U.32.0	Group subtract unsigned quadlets check overflow
G.SUB.U.64.0	Group subtract unsigned octlets check overflow
G.SUB.U.128.O	Group subtract unsigned hexlet check over.iow

Equivalencies

G.SET.E.Z.8	Group set equal zero bytes
GSET.E.Z.16	Group set equal zero doublets
GSET.E.Z.32	Group set equal zero quadlets
G.SET.E.Z.64	Group set equal zero octiets
G.SET.E.Z.128	Group set equal zero hexlet
GSET.GZ.8	Group set greater zero signed bytes
G.SET.G.Z.16	Group set greater zero signed doublets
G.SET.G.Z.32	Group set greater zero signed quadlets
G.SET.G.Z.64	Group set greater zero signed octlets
G.SET.G.Z.128	Group set greater zero signed hexlet
G.SET.GE.Z.8	Group set greater equal zero signed bytes
G.SET.GE Z.16	Group set greater equal zero signed doublets
G.SET.GE.Z.32	Group set greater equal zero signed quadlets
G.SET.GE Z.64	Group set greater equal zero signed octies
G.SET.GE.Z.128	Group set greater equal zero signed hexics
G.SET.L.Z.8	Group set less zero signed bytes
G.SET.LZ.16	Group set less zero signed doublets
G.SET.L.Z.32	Group set less zero signed gradiets
G.SET.LZ.64	Group set less zero signed octlets
G.SET.LZ.128	Group set less zero signed hexlet
G.SET.LE.Z.8	Group set less equal zero signed bytes
G.SET.LE.Z.16	Group set less equal zero signed doublets
GSET.LE.Z.32	Group set less equal zero signed quadlets
G.SET.LE.Z.64	Group set less equal zero signed octiets
G.SET.LE.Z.128	Group set less equal zero signed hexlet
G.SET.NE.Z.8	Group set not equal zero bytes
G.SET.NE.Z.16	Group set not equal zero doublets
G.SET.NE.Z.32	Group set not equal zero quadlets
G.SET.NE.Z.64	Group set not equal zero octlets
G SET.NE.2.128	Group set not equal zero hexlet
G.SET.LE.8	Group set less equal signed bytes
G.SET.LE.16	Group set less equal signed doublets
G.SET.LE.32	Group set less equal signed quadlets
G.SET.LE.64	Group set less equal signed octlets
G.SET.LE. 128	Group set less equal signed hexlet
G.SET.LE.U.8	Group set less equal unsigned bytes
G.SET.LE.U.16	Group set less equal unsigned apublets
G.SET.LE.U.32	Group set less equal unsigned quidlets
G.SET.LE.U.64	Group set icas equal unsigned orities
G.SET.LE.U.178	Group set less equal unsigned hexlet
G.SET.G.8	Group set signed greater bytes
G.SET.G.16	Group set signed greater doublets
G.SET.G.32	Group set signed greater quadlets
G.SET.G.64	Group set signed greater octiets
	and the Agent Acres of the

G.SET.G. 128	Group set signed greater hexlet	•
G.SET.G.U.8	Group set greater unsigned bytes	
G.SET.G.U.16	Group set greater unsigned doublets	
G.SET.G.U.32	Group set greater unsigned quadlets	
G.SET.G.U.64	Group set greater unsigned octlets	
G.SET.G.U.128	Group set greater unsigned heliet	

G.SET.E.Z.size rd=rc	←	G.SET.AND.E.size rd=rc,rc	
G.SET.G.Z.size rd=rc		G.SET.L.U.size rd=rc,rc	72
G.SET.GE.Z.size rd=rc		G.SET.GE.size rd=rc,rc	_
G.SET.L.Z.size rd=rc	=	G.SET.L.size rd=rc,rc	
G.SET.LE.Z.size rd=rc	(=	G.SET.GE.U.size rd=rc,rc	
G.SET.NE.Z.size rd=rc	←	G.SET.AND.NE.size rd=rc,rc	
G.SET.G.size rd=rb,rc	→	G.SET.L.size rd=rc,rb	
G.SET.G.U.size rd=rb,rc	· →	G.SET.L.U.size rd=rc,rb	
G.SET.LE.size rd=rb,rc	→	G.SET.GE.size rd=rc,rb	_
G.SET.LE.U.size rd=rb,rc	→	G.SET.GE.U.size rd=rc,rb	

Redundancies

⇔	G.SET rd	
⇔	G.ZERO rd	
⇔	G.ZERO rd	
⇔	G.ZERO rd	
<u>ن</u> ې	G.ZERO rd	
⇔	G.ZERO rd	
⇔	G.ZERO rd	
	### ##################################	⇔ G.ZERO rd ⇔ G.ZERO rd ⇔ G.ZERO rd

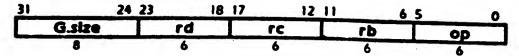
Selection

class	operation	cond	operand	size	check
arithmetic	SUB			8 16 32 64 128	
			NONE U	8 16 32 64 128	0
	SUB.L		NONE U	8 16 32 64 128	
boolean	SET.AND SET	E NE		8 16 32 64 128	
•	SET	L GE G LE	NONE U	8 16 32 64 128	
	SET	G GE L LE	Z	8 16 32 64 128	

Format

G.op.sizerd=rb,rc

rd=gopsize(rb,rc)



Description

Two values are taken from the contents of registers rc and rb. The specified operation is performed, and the result is placed in register rd.

Definition

```
def GroupReversed/op.size.rd.rc.rb/
     c - RegReadirc, 128).
     b ← RegRead(rb, 128)
     case op of
            G.SUB:
                  for i \leftarrow 0 to 128-size by size
                        divsize-1... ← Divsize-1... · Civsize-1...
                  endfor
           G.SUBL:
                 for i \leftarrow 0 to 128-size by size
                       t ← (Dissize-1 11 Dissize-1 il - (Cissize-1 ' Cissize-1 il
                        Assze-1_i ← l'size # (size-1) ? l'size || ($1$\frac{1}{2}$\frac{1}{2}$) : (size-1.0)
                 endfor
           G.SUBLU:
                 for i \leftarrow 0 to 128-size by size
                       t \leftarrow (0^1 \ 11 \ b_{\text{insize-1...il}} - (0^1 \ 11 \ c_{\text{insize-1...il}})
                       # size-1_i ← (tsize = 0) ? Osize: tsize-1.0
                 endfor
           G.SUB.O:
                 for i \leftarrow 0 to 128-size by size
                       t ← (Dissize-1 | 1 Dissize-1 ii - (Cissize-1 | 1 Cissize-1 ii
                       if flaire # taize-1) then
                             raise FixedPointArithmetic
                       endif
                       distre-1.1 ← lare-1.0
                 endfor
          G.SUB.U.O:
                 for i \leftarrow 0 to 128-size by size
                      t - 101 11 busize-1.1 - 101 /1 circize-1.1
                       if Rsize # 0) then
                             raise FixedPointArithmetic
                      endif
                      disize-1... ← lsize-1..0
                endfor
          G.SET.E:
```

```
for i \leftarrow 0 to 128-size by size \cdot
                    desize 1.1 ← |Desize-1.1 = Cieste-1.1size
              endfor
        G.SET.NE:
              for i ← 0 to 128-size by size
                    President + (Direstelli ≠ Chestellisise
              endfor
       GSETANDE:
             for i \leftarrow 0 to 128-size by size
                   G.SET.AND.NE:
             for i ← 0 to 128-rize by size
                   aissze-1..i ← ((Dissize-1..i and Cissize-1..i ≠ O)size
             endfor
       G.SET.L:
             for i ← 0 to 128-size by size
                   a_{i+size-1..i} \leftarrow (|rc = rb|, ? |b_{i+size-1..i} < 0| : |b_{i+size-1..i} < c_{i+size-1..i}|) size
             endfor
       G.SET.GE:
             for i \leftarrow 0 to 128-size by size
                  a_{i+size-1,j} \leftarrow ((rc = rb) ? (b_{i+size-1,j} \ge 0) : (b_{i+size-1,j} \ge c_{i+size-1,j}) size
            endfor
      GSET.LU:
            for i \leftarrow 0 to 128-size by size
                  dispize-1...i ← ((rc = rb) 7 (bissize-1..i > 0) :
                        (10 11 bissize-1...) < 10 11 cissize-1...)
            endfor
      G.SET.GE.U:
            for i \leftarrow 0 to 128-size by size
                  a_{i+size-1,i} \leftarrow (|rc = rb| ? |b_{i+size-1,i} \le 0):
                        (|0 | 1 | b<sub>i+size-1..i</sub>| ≥ |0 | 1 | c<sub>i+size-1..i</sub>|)|size
            endfor
endcase
RegWritefrd, 128, af
```

Exceptions

enddef

I red point anthmetic

Group Reversed Floating-point

These operations take two values from registers, perform a group of floating-point arithmetic operations on partitions of bits in the operands, and place the concatenated results in a register.

Operation codes

G.SET.E.F.16	Group set equal floating-point half
G.SET.E.F.16.X	Group set equal floating-point half exact
G.SET.E.F.32	Group set equal floating-point single
G.SET.E.F.32X	Group set equal floating-point single exact
G.SET.E.F.64	Group set equal floating-point double
G.SET.E.F.64.X	Group set equal floating-point double exact
G.SET.E.F. 128	Group set equal floating-point guad
G.SET.E.F. 128.X	Group set equal floating-point quad exact
G.SET.GE.F.16X	Group set greater equal floating-point half exact
G.SET.GE.F.32X	Group set greater equal floating-point single exact
G.SET.GE.F.64.X	Group set greater equal floating-point double exact
G.SET.GF.F.128.X	Group set greater equal floating-point quad exact
G.SET.LG.F.16	Group set less greater floating-point half
G.SET.LG.F.16.X	Group set less greater floating-point half exact
G.SET.LG.F.32	Group set less greater floating-point single
G.SET.LG.F.32.X	Group set less greater floating-point single exact
G.SET.LG.F.64	Group set less greater floating-point double
G.SET.LG.F.64.X	Group set less greater floating-point double exact
G.SET.LG.F.128	Group set less greater floating-point quad
G.SET.LG.F.128.X	Group set less greater floating-point quad exact
G.SET.L.F.16	Group set less floating-point half
G.SET.L.F. 16.X	Group set less floating-point half exact
G.SET.L.F.32	Group set less floating-point single
G.SET.LF.32.X	Group set less floating-point single exact
G.SET.LF.64	Group set less floating-point double
G.SET.LF.64.X	Group set less floating-point double exact
G.SET.L.F.128	Group set less floating-point quad
G.SET.LF.128.X	Group set less floating-point quad exact
G.SET.GE.F.16	Group set greater equal floating-point half
G.SET.GE.F.32	Group set greater equal floating-point single
G.SET.GE.F.64	Group set greater equal floating-point double
G.SET.GE.F.128	Group set greater equal floating-point quad
	and Acoust class woonid-houst dried

Equivalencies

G.SET.LE.F. 16X	Group set less equal floating-point half exact
G.SET.LE.F.32X	Group set less equal floating-point single exact
G.SET.LE.F.64.X	Group set less equal floating-point double exact
G.SET.LE.F. 128X	Group set less equal floating-point quad exact
G.SET.G.F.16	Group set greater floating-point half
G.SET.G.F. 16X	Group set greater floating-point half exact
G.SET.G.F.32	Group set greater floating-point single
G.SET.G.F.32X	Group set greater floating-point single exact
G.SET.G.F.64	Group set greater floating-point double
G.SET.G.F.64X	Group set greater floating-point double exact
G.SET.G.F. 128	Group set greater floating-point quad
G.SET.G.F. 128X	Group set greater floating-point quad exact
G.SET.LE.F. 16	Group set less equal floating-point half
G.SET.LE.F.32	Group set less equal floating-point single
G.SET.LE.F.64	Group set less equal floating-point double
G.SET.LE.F. 128	Group set less equal floating-point quad

G.SET.G.F.prec rd=rb,rc	→	G.SET.L.F.prec rd=rc,rb	
G.SET.G.F.prec.X rd=rb,rc	→	G.SET.L.F.prec.X rd=rc,rb	
G.SET.LE.F.prec rd=rb,rc	→	G.SET.GE.F.prec rd=rc,rb	
G.SET.LE.F.prec.X rd=rb,r.	→	G.SET.GE.F.prec.X rd=rc,rb	

Selection

class	ор	pre				round/trap	
sct	SET. E LG L GE G LE	16	32	64	128	NONE X	

Format

G.op.prec.round rd=rb,rc

rc=gopprecround(rb,ra)

31	24 23	18	17	12 11	6.5	0
G	.prec	rd	rc	rb	ОР	.round
	8	6	6	6		6

Description

The contents of registers ra and rb are combined using the specified floating-point operation. The result is placed in register rc. The operation is rounded using the specified rounding option or using round-to-nearest if not specified. If a rounding option is specified, the operation raises a floating-point exception if a floating-point invalid operation, divide by

zero, overflow, or underflow occurs, or when specified, if the result is inexact. If a rounding option is not specified, floating-point exceptions are not raised, and are handled according to the default rules of IEEE 754.

Definition

```
def GroupFloatingPointReversed(op.prec.round.rd.rc.rb) as
     c ← RegRead(rc, 128)
     b ← RegRead(rb, 128)
     for i \leftarrow 0 to 128-prec by prec
          ci ← Fiprec,cioprec-1.il
          bi ← Fiprecbiprec-1.il
          if round#NONE then
                if (dix = SNAN) or (ci.t = SNAN) then
                    raise FloatingPointArithmetic
                endif
                case op of
                     GSET.LF, G.SET.GEF:
                          if (di.t = ONAN) or (ci.t = ONAN) then
                               raise Floating?ointArithmetic
                     others: //nothing
               endcase
          endf
          case op of
               G.SET.L.F:
                    ai ← bi?≥ci
               G.SET.GE.F:
                    ai ← bil?cci
               G.SET.E.F:
                    ai ← be=ci
               G.SET.LG.F:
                    ai ← bi≠ci
          endcase
          g+bacc-1" ← St
     endfor
     RegWritefrd, 128, a)
enddef
```

Exceptions

l'loating-point arithmetic

Group Shift Left Immediate Add

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

GSHLIADD.8	Group shift left immediate add bytes
GSHLIADD.16	Group shift left immediate add doublets
GSHLIADD.32	Group shift left immediate add quadlets
GSHLIADD.64	Group shift left immediate add octlets
G.SHLIADD.128	Group shift left immediate add hexlet

Redundancies

G.SHLIADD.size rd=rd.rc,1	⇔ GAMsize rd@rc.rc

Format

G.op.sizerd=rc,rb,i

rd=gopsize(rc,rb,i)

31	24 23		18 17		12 11		6.5	21 0
G.sla	re .	rd		rc	-	rb	ashri	voo sh
8	-	6	700	6		6	6	2

assert 1≤i≤4 sh ← i-1

Description

The contents of registers rc and rb are partitioned into groups of operands of the size specified. Partitions of the contents of register rb are shifted left by the amount specified in the immediate field and added to partitions of the contents of register rc, yielding a group of results, each of which is the size specified. Overflows are ignored, and yield modular arithmetic results. The group of results is catenated and placed in register rd.

Definition

```
def GroupShiftLeftimmediateAdd(sh,size,ra,rb,rc)

C ← RegRead(rc, 128)

b ← RegRead(rb, 128)

for i ← 0 to 128-size by size

Aissize-1.i ← Cissize-1.i + (bissize-1-sh.i 11 01+sh)

endfor

RegWrite(rd, 128, a)
enddef
```

Tuc, Aug 17, 1999

Instruction Set Group Shift Left Immediate Add

Exceptions

none

- 180 -

MicroUnity

Group Shift Left Immediate Subtract

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

G.SHLI.SUB.8	Group shift left immediate subtract bytes
GSHLLSUB.16	Group shift left immediate subtract doublets
GSHLISUB.32	Group shift left immediate subtract quadlets
G.SHL.I.SUB.64	Group shift left immediate subtract octlets
G.SHL.I.SUB.128	Group shift left immediate subtract nexiet

Redundancies

G.SHL.I.SUB.size rd=rc,1,rc	⇔ G.COPY rd=rc
-----------------------------	----------------

Format

G.op.sizerd=rb,i,rc

rd=gopsize(rb,i,rc)

31	24 23	18 17		12 11	6.5	21 0
G.size		rd	rc	rb	GSHLISU	-Ish
8		6	6	6	6	2

assert 1≤i≤4 sh ← i-1

Description

The contents of registers re and rb are partitioned into groups of operands of the size specified. Partitions of the contents of register re are subtracted from partitions of the contents of register rb shifted left by the amount specified in the immediate field, yielding a group of results, each of which is the size specified. Overflows are ignored, and yield modular arithmetic results. The group of results is catenated and placed in register rd.

Definition

def GroupShiftLeftImmediateSubtract(sh,size,ra,rb,rc)

c ← RegRead(rc, 128)

b ← RegRead(rb, 128)

for i ← 0 to 128-size by size

āi-size-1...i ← [bi+size-1-sh...i 11 01+sh] - Ci+size-1..i

endfor

RegWrite(rd, 128, a)

enddef

Zeus System Architecture

Tue, Aug 17, 1999

Instruction Set Group Shift Left Immediate Subtract

Exceptions

0000

Group Subtract Halve

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

G.SUB.H.8.C	Group subtract halve signed bytes ceiling
G.SUB.H.8.F	Group subtract halve signed bytes floor
G.SUB.H.8.N	Group subtract halve signed bytes nearest
G.SUB.H.8.Z	Group subtract halve signed bytes zero
G.SUB.H.16.C	Group subtract halve signed doublets ceiling
G.SU9.H.16.F	Group subtract halve signed doublets floor
G.SUB.H.16.N	Group subtract halve signed doublets nearest
G.SUB.H. 16.Z	Group subtract halve signed doublets zero
G.SUB.H.32.C	Group subtract halve signed quadlets ceiling
G.SUB.H.32.F	Group subtract halve signed quadlets floor
G.SUB.H.32.N	Group subtract halve signed quadlets nearest
G.SUB.H.32.Z	Group subtract halve signed quadlets zero
G.SUB.H.64.C	Group subtract halve signed octlets ceiling
G.SUB.H.64.F	Group subtract halve signed octlets floor
G.SUB.H.64.N	Group subtract halve signed octlets nearest
G.SUB.H.64.Z	Group subtract halve signed octlets zero
G.SUB.H.128.C	Group subtract halve signed hexlet ceiling
G.SUB.H.128.F	Group subtract halve signed hexlet floor
G.SUB.H.128.N	Group subtract halve signed hexlet nearest
G.SUB.H.128.Z	Group subtract halve signed hexlet zero
G.SUB.H.U.8.C	Group subtract halve unsigned bytes ceiling
G.SUB.H.U.8.F	Group subtract halve unsigned bytes floor
G.SUB.H.U.8.N	Group subtract halve unsigned bytes nearest
G.SUB.H.U.8.Z	Group subtract halve unsigned bytes zero
G.SUB.H.U. 16.C	Group subtract halve unsigned doublets ceiling
G.SUB.H.U. 16.F	Group subtract halve unsigned doublets floor
G.SUB.H.U.16.N	Group subtract halve unsigned doublets nearest
G.SUB.H.U.16.Z	Group subtract halve unsigned doublets zero
G.SUB.H.U.32.C	Group subtract halve unsigned quadlets ceiling
G.SUB.H.U.32.F	Group subtract halve unsigned quadlets floor
G.SUB.H.U.32.N	Group subtract halve unsigned quadlets nearest
G.SUB.H.U.32.Z	Group subtract halve unsigned quadlets zero
G.SUB.H.U.64.C	Group subtract halve unsigned octlets ceiling
G.SUB.H.U.64.F	Group subtract halve unsigned octlets floor
G.SUB.H.U.64.N	Group subtract halve unsigned octlets nearest
G.SUB.H.U.64.Z	Group subtract halve unsigned octlets zero
G.SUB.H.U. 128.C	Group subtract halve unsigned hexlet ceiling
G.SUB.H.U. 128.F	Group subtract halve unsigned hexlet floor
G.SUB.H.U.128.N	Group subtract halve unsigned hexlet nearest

G.SUB.H.U. 128.Z	Group subtract halve unsigned hexlet zero

Redundancies

G.SUB.H.size.rnd rd=rc,rc	⇔	GZERO rd	
G.SUB.H.U.size.rnd rd=rc,rc	⇔	GZERO rd	

Format

G.op.size.rnd rd=rb.rc

rd=gopsizerna(rb,rc)

31	24	23	18 17	12	11	6 5	21 0
G.sl	re	rd		rc	rb	ot	rnd
8		6		6	6	4	2

Description

The contents of registers re and rb are partitioned into groups of operands of the size specified and subtracted, halved, rounded and limited as specified, yielding a group of results, each of which is the size specified. The group of results is catenated and placed in register rd.

The result of this operation is always signed, whether the operands are signed or unsigned.

Definition

```
def GroupSubtractHalve(op;rnd,size,rd,rc,rb)
      c ← RegRead(rc, 128)
      b ← RegRead(rb, 128)
      case op of
            G.SUB.H.C. G.SUB.H.F. G.SUB.H.N. G.SUB.H.Z:
                   as \leftarrow cs \leftarrow bs \leftarrow 1
            G.SUB.H.U.C. G.SUB.H.U.F. G.SUB.H.U.N. G.SUB.H.U.Z
                   as \leftarrow 1
                   cs \leftarrow bs \leftarrow 0
     endcase
     for i \leftarrow 0 to 128-size by size
            p \leftarrow \{(bs \text{ and } b_{size-1}) \mid 1 \mid b_{size-1+i,.i}\} - \{(cs \text{ and } c_{size-1}) \mid 1 \mid c_{size-1+i,.i}\}
            case md of
                  none. N:
                         s ← Osize II -pi
                  Z:
                         s ← Osize 11 Psize
                  F:
                         s \leftarrow osize+1
                  C:
                        s ← Osize | | | | 1
           endcase
```

```
v ← (las & p<sub>size</sub>) | |p| + (01|s)

if v<sub>size+1</sub> = (as & v<sub>size</sub>) then

a<sub>size+1+Li</sub> ← v<sub>size+1</sub>

else

a<sub>size+1+Li</sub> ← as ? (v<sub>size+1</sub> | 1 -v<sub>size+1</sub>) : 1<sup>size</sup>

endif

endif

endifor

RegWrite(rd. 128, a)

enddef
```

Exceptions

none

Group Ternary

These operations take three values from registers, perform a group of calculations on partitions of bits of the operands and place the catenated results in a fourth register.

Operation codes

the same of the sa		
G.MUX	Group multiplex	

Redundancies

G.MUX ra=rd,rc,rc	⇔	G.COPY ra=ic
G.MUX ra=ra,rc,rb	⇔	G.BOOLEAN ra@rc,rb,0x11001010
G.MUX ra=rd,ra,rb	⇔	G.BOOLEAN ra@rd,rb,0x11100010
G.MUX ra=rd,rc,ra	⇔	G.BOOLEAN ra@rd,rc,0x11011000
G.MUX ra=rd,rd,rb		G.OR ra=rd,rb
G.MUX ra=rd,rc,rd		GAND ra=rd,rc

Format

G.MUX ra=rd,rc,rb

ra=gmux(rd,rc,rb)

31	24	23	18 17	12	11	6.5	0
G.M	UX	rd		rc	rb		
8		6		6	6		

Description

The contents of registers rd, rc, and rb are fetched. Each bit of the result is equal to the corresponding bit of rc, if the corresponding bit of rd is set, otherwise it is the corresponding bit of rb. The result is placed into register ra.

Definition

```
del GroupTernarylop,size,rd,rr,rb,raj as
d ← RegReadfrd, 128j
c ← RegReadfrb, 128j
b ← RegReadfrb, 128j
case op of
G.MUX:
```

a ← (c and d) or (b and not d) endcase RegWrite(ra, 128, a) endde(

Exceptions

none

Crossbar

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

X.COMPRESS.2	Crossbar compress signed pecks
X.COMPRESS.4	Crossbar compress signed nibbles
X.COMPRESS.8	Crossbar compress signed bytes
X.COMPRESS. 16	Crossbar compress signed doublets
X.COMPRESS.32	Crossbar compress signed quadlets
X.COMPRESS.64	Crossbar compress signed octlets
X.COMPRESS.128	Crossbar compress signed hexlet
X.COMPRESS.U.2	Crossbar compress unsigned packs
X.COMPRESS.U.4	Crossbar compress unsigned nibbles
X.COMPRESS.U.8	Crossbar compress unsigned bytes
X.COMPRESS.U.16	Crossbar compress unsigned doublets
X.COMPRESS.U.32	Crossbar compress unsigned quadlets
X.COMPRESS.U.64	Crossbar compress unsigned outlets
X.COMPRESS.U. 128	Crossbar compress unsigned heidet
X.EXPAND.2	Crossbar expand signed pecks
X.EXPAND.4	Crossbar expand signed nibbles
X.EXPAND.8	Crossbar expand signed bytes
X.EXPAND.16	Crossbar expand signed doublets
XEXPAND.32	Crossbar expand signed quadlets
X.EXPAND.64	Crossbar expand signed octlets
XEXPAND.128	Crossbar expand signed hexlet
X.EXPAND.U.2	Crossbar expand unsigned pecks
X.EXPAND.U.4	Crossbar expand unsigned nibbles
X.EXPAND.U.8	Crossbar expand unsigned bytes
X.EXPAND.U.16	Crossbar expand unsigned doublets
X.EXPAND.U.32	Crossbar expand unsigned quadlets
X.EXPAND.U.64	Crossbar expand unsigned octlets
X.EXPAND.U.128	Crossbar expand unsigned hexlet
X.ROTL2	Crossbar rotate left pecks
XROTL4	Crossbar rotate left nibbles
X.ROTL8	Crossbar rotate left bytes
X.ROTL16	Crossbar rotate left doublets
X.ROTL32	Crossbar rotate left quadlets
XROTL64	Crassbar rotate left octlets
XROTL128	Crossbar rotate left hexlet
X.ROTR.2	Crossbar rotate right pecks
X.ROTR.4	Crossbar rotate right nibbles
X.ROTR.6	Crossbar rotate right bytes
X.ROTR.16	Crossbar rotate right doublets
X.ROTR.32	Crossbar rotate right quadlets

X.ROTR.64	Crossbar rotate right octlets
X.ROTR.128	Crossbar rotate right hexlet
XSHL2	Crossbar shift left pecks
X.SHL.2.O	Crossbar shift left signed pecks check overflow
XSHL4	Crossbar shift left nibbles
XSHL4.O	Crossbar shift left signed nibbles check overflow
XSHL8	Crossbar shift left bytes
XSHL8.O	Crossbar shift left signed bytes check overflow
XSHL16	Crossbar shift left doublets
XSHL16.0	Crossbar shift left signed doublets check overflow
XSHL3?	Crossbar shift left quadlets
XSHL32.O	Crossbar shift left signed quadlets check overflow
X.SHL.64	Crossbar shift left octlets
XSHL64.O	Crossbar shift left signed octlets check overflow
XSHL 128	Crossbar shift left hexlet
XSHL 128.0	Crossbar shift left signed hexlet check overflow
XSHLU.2.O	Crossbar shift left unsigned pecks check overflow
X.SHL.U.4.O	Crossbar shift left unsigned nibbles check overflow
X.SHLU.8.O	Crossbar shift left unsigned bytes check overflow
XSHLU.16.0	Crossbar shift left unsigned doublets check overflow
XSHLU.32.0	Crossbar shift left unsigned quadlets check overflow
XSHLU.64.0	Crossbar shift left unsigned octlets check overflow
XSHLU.128.O	Crossbar shift left unsigned hexlet check overflow
X.SHR.2	Crossbar signed shift right pecks
X.SHR.4	Crossbar signed shift right nibbles
X.SHR.8	Crossbar signed shift right bytes
X.SHR.16	Crossbar signed shift right doublets
X.SHR.32	Crossbar signed shift right quadlets
X.SHR.64	Crossbar signed shift right octlets
X.SHR.128	Crossbar signed shift right hexlet
X.SHR.U.2	Crossbar shift right unsigned pecks
X.SHR.U.4	Crossbar shift right unsigned nibbles
X.SHR.U.8	Crossbar shift right unsigned bytes
X.SHR.U.16	Crossbar shift right unsigned doublets
X.SHR.U.32	Crossbar shift right unsigned quadlets
X.SHR.U.64	Crossba shift right unsigned octlets
X.SHR.U.128	Crossbar shift right unsigned hexlet

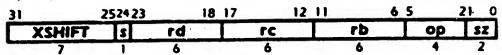
Selection

class	ОР		size	
precision	EXPAND COMPRESS	EXPAND.U COMPRESS.U	2 4 8 16	32 64 128
shift	ROTR ROTL SHLO SHLU.	SHR SHL O SHR.U	2 4 8 16	32 64 128

Format

Xop. vize rd=rc.rb

rd=xopsize(rc.rb)



```
lsize ← log(size)
s ← lsize2
sz ← lsize1_0
```

Description

Two values are taken from the contents of registers re and rb. The specified operation is performed, and the result is placed in register rd.

Definition

```
def Crossbarlop, size, rd, rc, rbj
      c - RegRead(rc, 128)
      b ← RegRead(rb, 128)
      shift - b and (size-1)
      case op<sub>5...2</sub> 11 0<sup>2</sup> of
            X.COMPRESS:
                   hsize ← size/2
                   for i ← 0 to 64-hsize by hsize
                          if shift ≤ hsize then
                                 Biothsize-1... ( Cioioshiftohsize-1...ioioshift
                          else
                                antsize-1..i ← cshift-hsize 11 Cininsize-1..ininshift
                          endif
                   endfor
                   a_{127.64} \leftarrow 0
            X.COMPRESS.U:
                   hsize \leftarrow size/2
                   for i \leftarrow 0 to 64-hsize by hsize
                          if shift ≤ hsize then
                                annsize-1..i ← Cioinshift-Insize-1..ionshift
                                annsize-1.1 - Oshift-hsize | | Chinsize-1...hinshift
                          endil
                   endfor
                   a127.64 ← 0
             XEXPAND:
                   hsize \leftarrow size/2
                   for i - 0 to 64-hsize by hsize
                          if shift ≤ hsize then
                                a_{\text{prinsize-1..in}} \leftarrow c_{\text{physice-1}}^{\text{hysize-shift}} | | C<sub>i-hysize-1..i</sub> | | | Oshift
```

enddef

```
· else
                       divisize-1..ivi ← Civaze-shift-1..i 11 Oshift
                 endif
           endfor
     XEXPAND.U:
           hsize \leftarrow size/2
           for i 

0 to 64-hsize by hsize
                 if shift s hsize then
                       Siviosize-1...ii ← Ohsize-shift | 1 Conside-1... | 1 Oshift
                 else
                       diviosize-1_ini ← Ciosize-shift-1,: 11 Oshift
                 endif
           endfor
     XROTL:
           for i ← 0 to 128-size by size
                 apraze-1...i ← Circize-1-shift.i 11 Circize-1.irsize-1-shift
           .endfor
     X.ROTR:
           for i \leftarrow 0 to 128-size by size
                 ajosize-1_i ← Cjoshift-1_i 11 Cjosize-1_joshift
           endfor
     X.SHL:
           for 1 += 0 to 128-size by size
                 Styrize-1... ← Cysize-1-shift.i 11 Oshift
           endfor
     X.SHLO:
           for i \leftarrow 0 to 128-size by size
                 if Civaze-1, ivaze-1-shift # Civaze-1-shift then
                       raise FixedPointArithmetic
                 ajusize-1..i ← Cjusize-1-shift.i | 1 Oshift
           endfor
     X.SHLU.O:
           for i \leftarrow 0 to 128-size by size
                 if Ciosize-1.iosze-shift # Oshift then
                       raise FixedPointArithmetic
                 Sheise-1... + Chese-1-shift... | Oshift
           endfor
     X.SHR:
           for i \leftarrow 0 to 128-size by size
                 Sporse-1... ← CShift | | | Cipsize-1...ipshift
           endfor
     X.SHR.U:
           for i \leftarrow 0 to 128-size by size
                 ajusize-1...i ← Oshift | | Cjusize-1...iushift
           endfor
endcase
RegWrite(rd. 128, a)
```

Instruction Set Consular

Exceptions

Fixed point anthmetic

Crossbar Extract

These operations take operands from three registers, perform operations on partitions of bits in the operands, and place the concatenated results in a fourth register.

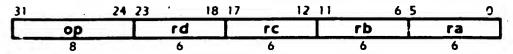
Operation codes

X.EXTRACT	Crossbar extract	
-----------	------------------	--

Format

X.EX.TACT ra=rd.rc.rb

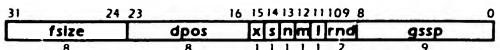
ra=xextract(rd,rc,rb)



Description

The contents of registers rd, rc, and rb are fetched. The specified operation is performed on these operands. The result is placed into register ra.

Bits 31.0 of the contents of register rb specifies several parameters which control the manner in which data is extracted, and for certain operations, the manner in which the operation is performed. The position of the control fields allows for the source position to be added to a fixed control value for dynamic computation, and allows for the lower 16 bits of the control field to be set for some of the simpler extract cases by a single GCOPYI.128 instruction. The control fields are further arranged so that if only the low order 8 bits are non-zero, a 128-bit extraction with truncation and no rounding is performed.



The table below describes the meaning of each label:

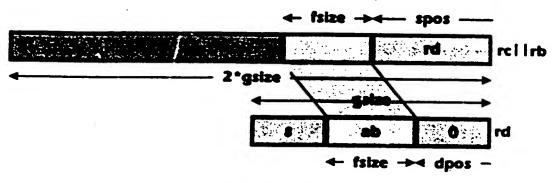
label	bits	meaning		
fsize	8	field size		
dpos	8	destination position		
λ	1	reserved		
S	1	signed vs. unsigned		
n	1	reserved		
m	1	merge vs. extract		
1	1	reserved		
rnd	2	reserved		
gssp	9	group size and source position		

The 9-bit gssp field encodes both the group size, gsize, and source position, spos, according to the formula gssp = 512-4*gsize+spos. The group size, gsize, is a power of two in the range 1...128. The source position, spos, is in the range 0...(2*gsize)-1.

The values in the s, n, m, l, and md fields have the following meaning:

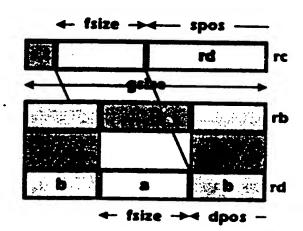
values	S	n	m	1	rnd
0	unsigned		extract		
1	signed		merge		
2					
3					

For the X.EXTRACT instruction, when m=0, the parameters are interpreted to select a fields from the catenated contents of registers rd and rc, extracting values which are catenated and placed in register ra.:



Crossbar extract

For a crossbar-merge-extract (X.EXTRACT when m=1), the parameters are interpreted to merge a fields from the contents of register rd with the contents of register rc. The results are catenated and placed in register ra.



Crossbar merge extract

Definition

```
del CrossbarExtractiop,ra,rb,rc.rd) as
               d - RegReadtrd, 128)
               c ← RegRead(rc, 128)
               b ← RegRead(rb, 128)
               case b<sub>8.0</sub> of
                               0..255:
                                                gsize ← 128
                               256..383:
                                               gsize ← 64
                               384_447:
                                                gsize \leftarrow 32
                               448..479:
                                                gsize ← 16
                               480..495:
                                                gsize \leftarrow 8
                                496 .503:
                                                gsize ← 4
                                504..507:
                                                gsize \leftarrow 2
                                508.511:
                                                gsze ← 1
                endcase
                m ← b12
                as ← signed ← 514
                h ← (2-ml*qsize
                 spos \leftarrow (b<sub>B.O</sub>) and ((2-m)*gsize-1)
                dpos ← (0 11 b23..16) and (gsize-1)
                 state \leftarrow (0.11 \text{ b}_{31..24}) and [gsize-1]
                 the first f(x) = 0 or (f(x) = 0) or (f(x
                fsize - (tfsize + spos > h) ? h - spos : tfsize :
                for i \leftarrow 0 to 128-gaze by gaze
                                case op of
                                                X.EXTRACT:
                                                                if in then
                                                                                p \leftarrow d_{gsize+-1..i}
                                                                                p \leftarrow \{d \mid l \mid C|_2 \leq size + i\} - 1...2^si
                                                                endif
                                endcase
                                v \leftarrow \{as \& p_{n-1}\} | p
                                w ← (as & v<sub>spos+fsize-1</sub>)gsize-fsize-dpos | | V<sub>fsize-1+spos.spos</sub> | | O<sup>dpos</sup>
                                if m then
                                                asze-1+1.1 ← Cgsze-1+i.dpos+fsize+i 11 Wapos+fsize-1..dpos 11 Cdpos-1+1..i
                                eke
                                                 asise-1+1 ← M
                           . endif
                 endfor
                 RegWrite(ra, 128, a)
  enddef
```

Exceptions

none

MicroUnity

Crossbar Field

These operations take operands from a register and two immediate values, perform operations on partitions of bits in the operands, and place the concatenated results in the second register.

Operation codes

X.DEPOSIT.2	Crossbar deposit signed pecks
X.DEPOSIT.4	Crossbar deposit signed nibbles
X.DEPOSIT.8	Crossbar deposit signed bytes
X.DEPOSIT. 16	Crossbar deposit signed doublets
X.DEPOSIT.32	Crossbar deposit signed quadlets
X.DEPOSIT.64	Crossbar deposit signed octlets
X.DEPOSIT.128	Crossbar deposit signed hexlet
X.DEPOSIT.U.2	Crossbar deposit unsigned pecks
X.DEFOSIT.U.4	Crossbar deposit unsigned nibbles
X.DEPOSIT.U.8	Crossbar deposit unsigned bytes
X.DEPOSIT.U.16	Crossbar deposit unsigned doublets
X.DEPOSIT.U.32	Crossbar deposit unsigned quadlets
X.DEPOSIT.U.64	Crossbar deposit unsigned octlets
X.DEPOSIT.U.128	Crossbar deposit unsigned hexlet
X.WITHDRAW.U.2	Crossbar withdraw unsigned pecks
X.WITHDRAW.U.4	Crossbar withdraw unsigned nibbles
X.WITHDRAW.U.8	Cross bar withdraw unsigned bytes
X.WITHDRAW.U.16	Crossbar withdraw unsigned doublets
X.WITHDRAW.U.32	Crossbar withdraw unsigned quadlets
X.WITHDRAW.U.64	Crossbar withdraw unsigned octlets
X.WITHDRAW.U.128	Crossbar withdrav: unsigned hexlet
X.V/ITHDRAW.2	Crossbar withdraw pecks
X.WITHDRAW.4	Crossbar withdraw nibbles
X.WITHDRAW.8	Crossbar withdraw bytes
X.WITHDRAW.16	Crossbar withdraw doublets
X.WITHDRAW.32	Crossbar withdraw quadlets
X.WITHDRAW.64	Crossbar withdraw octlets
X.WTHDRAW.128	Crossbar withdraw hexlet

Equivalencies

XSEXL2	Crossbar extend immediate signed pecks
X.SEX 1.4	Crossbar extend immediate signed nibbles
XSEX18	Crossbar extend inmediate signed bytes
XSEXL16	Crossbar extend immediate signed doublets
XSEX132	Crossbar extend immediate signed quadlets
XSEX1.64	Crossbar extend immediate signed octlets
XSEX1.128	Crossbar extend immediate signed hexlet
XZEX.1.2	Crossbar extend immediate unsigned pecks
XZEX1.4	Crossbar extend immediate unsigned nibbles
XZEXI.8	Crossbar extend immediate unsigned bytes
X.ZEXI.16	Crossbar extend immediate unsigned doublets
X.ZEX.1.32	Crossbar extend immediate unsigned quadlets
X.ZEX1.64	Crossbar extend immediate unsigned octlets
X.ZEX.1.128	Crossbar extend immediate unsigned hexlet

XSHLI.gsize rd=rc,i	→ X.DEPOSIT.gsize rd=rc,size-i,i
X.SHR.I.gsize rd=rc,i	→ X.WITHDRAW.gsize rd=rc,size-i,i
X.SHRU.I.gsize rd=rc,i	→ X.WITHDRAW.U.gsize rd=rc,size-i,i
X.SEX.I.gsize rd=rc,i	→ X.DEPOSIT.gsize rd=rc.i,0
XZEX.I.gsize rd=rc,i	→ X.DEPOSIT.U.gsize rd=rc,i,0

Redundancies

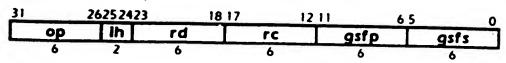
X.DEPOSIT.gsize rd=rc.gsize,0	⇔	X.COPY rd=rc	
X.DEPCSIT.U.gsize rd=rc,gsize,0	₩	X.COPY rd=rc	
XWITHDRAW.gsize rd=rc,gsize,0	⇔	X.COPY rd=rc	
X.WITHDRAW.U.gsize rd=rc,gsize,0	_	X.COPY rd=rc	

Format

X.op.gsize

rd=rc,isize,ishift

rd=xopgsize(rc,isize,ishift)



assert isize+ishift ≤ gsize

assert isize≥1

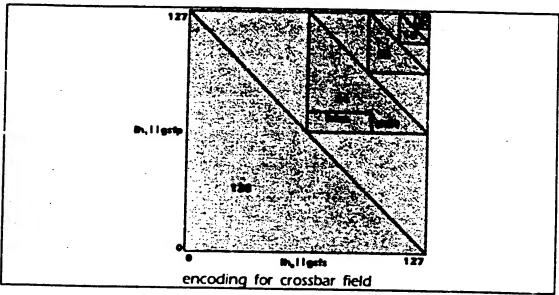
iho 11 gsfs ← 128-gsize+isize-1

ih1 11 gsfp ← 128-gsize+ishift

Description

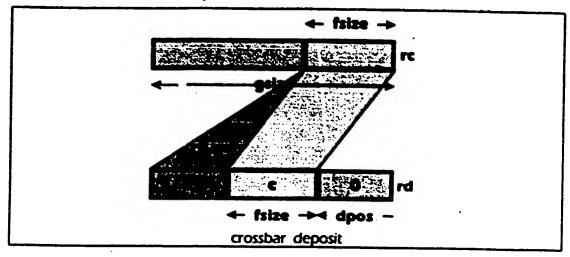
The contents of register rc is fetched, and 7-bit immediate values are taken from the 2-bit ih and the 6-bit gsfp and gsfs fields. The specified operation is performed on these operands. The result is placed into register rd.

The diagram below shows legal values for the ih, gsfp and gsfs fields, indicating the group size to which they apply.

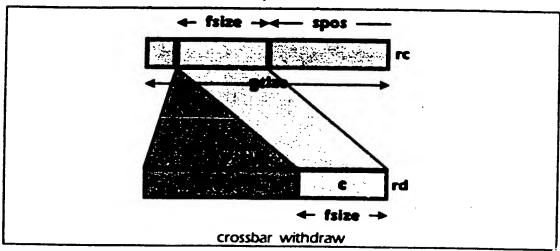


The ih, gsfp and gsfs fields encode three values: the group size, the field size, and a shift amount. The shift amount can also be considered to be the source bit field position for group-withdraw instructions or the destination bit field position for group-deposit instructions. The encoding is designed so that combining the gsfp and gsfs fields with a bitwise-and produces a result which can be decoded to the group size, and so the field size and shift amount can be easily decoded once the group size has been determined.

The crossbar-deposit instructions deposit a bit field from the lower bits of each group partition of the source to a specified bit position in the result. The value is either sign-extended or zero-extended, as specified.



The crossbar-withdraw instructions withdraw a bit field from a specified bit position in the each group partition of the source and place it in the lower bits in the result. The value is either sign-extended or zero-extended, as specified.



Definition

def CrossbarField|op.rd,rc,gsfp,gsfs| as

c ← RegRead|rc, 128|
case ||op₁ || 1 gsfp| and |op₀ || 1 gsfs|| of
0.63:

gsze ← 128
64..95:
gsze ← 64

```
96..111:
                   gsize ← 32
             112..119:
                   gsze ← 16
             120..123:
                   gaze - 8
             124..125:
                  gsize ← 4
             126:
                  gsize \leftarrow 2
             127:
                  raise ReservedInstruction
       endcase
       ishift - (op; 11 gsfp) and (gsize-1)
       raze ← ((opo 11 gsfs) and (gsize-1))+1
       # (ISTART+ISIZE>QSIZE)
            raise Reservedinstruction
       endd
      case op of
            X.DEPOSIT:
                 for i ← 0 to 128-gsize by gsize
                       angsize-1... ← Chisze-ishift | | Cinisize-1... | | Oishift
                 endfor
            X.DEPOSIT.U:
                 for i \leftarrow 0 to 128-gaze by gaze
                       anguze-1... ← Ogsize-isize-ishift || Cinisize-1... || Oishift
                 endfor
           XWITHDRAW:
                 for i \leftarrow 0 to 128-gaze by gaize
                      Auguze-1... ← Cuze-isize
Historishift-1 11 Cuisize-ishift-1..i-ishift
                 endfor
           X.WITHDRAW.U:
                 for i \leftarrow 0 to 126-gaze by gaze
                      anguze-1 i ← Ogsize-isize | | Cinsize-ishift-1..inishift
                 endfor
     endcase
     RegWritefrd, 128, al
enddef
```

Exceptions

Reserved instruction

Crossbar Field Inplace

These operations take operands from two registers and two immediate values, perform operations on partitions of bits in the operands, and place the concatenated results in the second register.

Operation codes

X.DEPOSIT.M.2	Crossbar deposit merge pecks	
X.DEPOSIT.M.4	Crossbar deposit merge nibbles	
X.DEPOSIT.M.8	Crossbar deposit merge bytes	
X.DEPOSIT.M.16	Crossbar deposit merge doublets	·
X.DEPOSIT.M.32	Crossbar deposit merge quadlets	
X.DEPOSIT.M.64	Crossbar deposit merge octlets	
X.DEPOSIT.M.128	Crossbar deposit merge hexlet	

Equivalencies

X.DEPOSIT.M.1	Crossbar deposit mer	ge bits
X.DEPOSIT.M.1 rd@re	1.0 - XCOP	Y rd=rc

Redundancies

The state of the s	
X.DEPOSIT.M.qsize rd@rc,qsize,0	VCORV and an
INDEPOSIT.MEDIZE TOGICOSZEO	⇔ XCOPY rd=rc

Format

X.op.qsize

rd@rc.isize,ishift

rd=xopgsize(rd,rc,isize,ishift)

31	262524	23	18 17	1;	2 1 1	6 5	0
Ор	Ih	rd		rc	gsfp		gsfs
6)	6		4	6		_

assert isize+ishift ≤ gsize

assert isize≥1

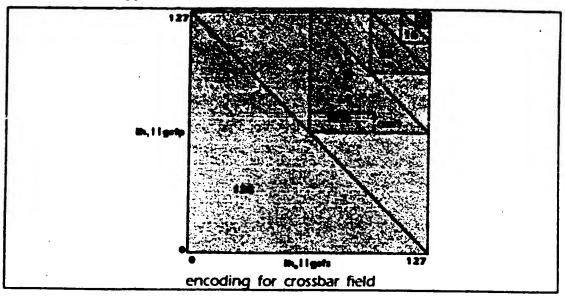
ino 11 gsfs ← 128-gsize+isize-1

ih 11 gsfp \leftarrow 128-gsize+ishift

Description

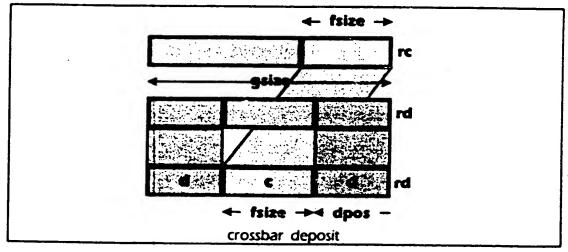
The contents of registers rd and rc are fetched, and 7-bit immediate values are taken from the 2-bit ih and the 6-bit gsfp and gsfs fields. The specified operation is performed on these operands. The result is placed into register rd.

The diagram below shows legal values for the ih, gsfp and gsfs fields, indicating the group size to which they apply.



The ih, gsfp and gsfs fields encode three values: the group size, the field size, and a shift amount. The shift amount can also be considered to be the source bit field position for group-withdraw instructions or the destination bit field position for group-deposit instructions. The encoding is designed so that combining the gsfp and gsfs fields with a bitwise-and produces a result which can be decoded to the group size, and so the field size and shift amount can be easily decoded once the group size has been determined.

The crossbar-deposit-merge instructions deposit a bit field from the lower bits of each group partition of the source to a specified bit position in the result. The value is merged with the contents of register rd at bit positions above and below the deposited bit field. No sign- or zero-extension is performed by this instruction.



Definition

```
def CrossbarFieldInplace(द्याः ट. १८ वर्षाय, वर्या, वर्या, वर्या, वर्या, वर्षाय, वर्या, वर्या, वर्या, वर्या, वर्या, व
                            c ← RegRead(rc, 128)
                            d - RegReadird, 128)
                            case (top; 11 gsfp) and topo 11 gsfs)) of
                                                    0..63:
                                                                           gsize ← 128
                                                    64..95:
                                                                           gsize ← 64
                                                    96..111:
                                                                           gsize ← 32
                                                    112..119:
                                                                           gsize ← 16
                                                     120..123:
                                                                           gsize ← 8
                                                    124..125:
                                                                          gsize ← 4
                                                    126:
                                                                        gsze ← 2
                                                    127:
                                                                         raise ReservedInstruction
                          endcase
                          ishift ← [op; 11 gsfp] and [gsize-1]
                          isize \leftarrow ||opo | | gsfs| and |gsize-1||+1
                         if [ishitt+isize>gsize]
                                                raise ReservedInstruction
                       for i \leftarrow 0 to 128-gsize by gsize
                                                angsize-1..i ← dingsize-1..inisizenishift 11 Cinisize-1..i 11 dinishift-1..i
                       endfor
                       RegWritefrd, 128, a)
enddef
```

Exceptions

Reserved instruction

Crossbar Inplace

These operations take operands from three registers, perform operations on partitions of bits in the operands, and place the concatenated results in the third register.

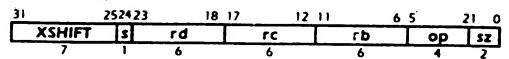
Operation codes

XSHLM.2	Crossbar shift left merge pecks	
XSHLM.4	Crossbar shift left merge nibbles	
XSHLM.8	Crossbar shift left merge bytes	-
XSHLM.16	Crossbar shift left merge doublets	
XSHLM.32	Crossbar shift left merge quadlets	
XSHLM.64	Crossbar shift left merge octlets	
XSHLM.128	Crossbar shift left merge hexlet	
X.SHR.M.2	Crossbar shift right merge pecks	
X.SHR.M.4	Crossbar shift right merge nibbles	
X.SHR.M.8	Crossbar shift right merge bytes	
X.SHR.M.16	Crossbar shift right merge doublets	-
X.SHR.M.32	Crossbar shift right merge quadlets	
X.SHR.M.64	Crossbar shift right merge octlets	
X.SHR.M.128	Crossbar shift right merge hexlet	

Format

Xop.size rd@rc,rb

rd=xopsize(rd,rc,rb)



Isize ← log(size)

 $s \leftarrow lsize_2$

 $sz \leftarrow lsize_{1..0}$

Description

The contents of registers rd, rc and rb are fetched. The specified operation is performed on these operands. The result is placed into register rd.

Register rd is both a source and destination of this instruction.

Definition

def Crossbartnplace(op,size,rd,rc,rb) as

d ← RegReadfrd, 128)

c - RegReadirc, 1281

nous

Crossbar Short Immediate

These operations take operands from a register and a short immediate value, perform operations on partitions of bits in the operands, and place the concatenated results in a register.

Operation codes

X.COMPRESS.I.2	Crossbar compress immediate signed pecks
X.COMPRESS.I.4	Crossbar compress immediate signed nibbles
X.COMPRESS.I.8	Crossbar compress immediate signed bytes
X.COMPRESS.I. 16	Crossbar compress immediate signed doublets
X.COMPRESS.1.32	Crossbar compress immediate signed quadlets
X.COMPRESS.I.64	Crossbar compress immediate signed octlets
X.COMPRESS.I.128	Crossbar compress immediate signed heidet
X.COMPRESS.I.U.2	Crossbar compress immediate unsigned pecks
X.COMPRESS.I.U.4	Crossbar compress immediate unsigned nibbles
X.COMPRESS.I.U.8	Crossbar compress immediate unsigned bytes
X.COMPRESS.I.U.16	Crossbar compress immediate unsigned doublets
X.COMPRESS.I.U.32	Crossbar compress immediate unsigned quadlets
X.COMPRESS.I.U.64	Crossbar compress immediate unsigned octlets
X.COMPRESS.I.U.128	Crossbar compress immediate unsigned heidet
X.EXPAND.1.2	Crossbar expand immediate signed pecks
X.EXPAND.I.4	Crossbar expand immediate signed nitibles
X.EXPAND.1.8	Crossbar expand immediate signed bytes
X.EXPAND.I.16	Crossbar expand immediate signed doublets
X.EXPAND.1.32	Crossbar expand immediate signed quadlets
X.EXPAND.1.64	Crossbar expand immediate signed octlets
XEXPAND.I.128	Crossia: expand immediate signed hexlet
X.EXPAND.1.U.2	Crossoar expand immediate unsigned pecks
X.EXPAND.I.U.4	Crosspar expand immediate unsigned nibbles
X.EXPAND.I.U.8	Crossbar expand immediate unsigned bytes
X.EXPAND.I.U.16	Crossbar expand immediate unsigned doublets
X.EXPAND.I.U.32	Crossbar expand immediate unsigned quadlets
X.EXPAND.I.U.64	Crossbar expand immediate unsigned octlets
X.EXPAND.I.U.128	Crossbar expand immediate unsigned hextet
X.ROTL.I.2	Crossbar rocate left immediate pecks
X.ROTL1.4	Crossbar rotate left immediate nibbles
X.ROTL.I.8	Crossbar rotate left immediate bytes
X.ROTL.I.16	Crossbar rotate left immediate doublets
X.ROTL.I.32	Crossbar rotate left immediate quadlets
X.ROTL1.64	Crossbar rotate left immediate octlets
XROTLI.128 /	Crossbar rotate left immediate hexlet
X.ROTR.I.2	Crossbar rotate right immediate pecks
X.ROTR.I.4	Crossbar rotate right immediate nibbles
X.ROTR.I.8	Crossbar rotate right immediate bytes
X.ROTR.I.16	Crossbar rotate right immediate doublets

X.ROTR.1.32 Crossbar rotate right immediate ocdets X.ROTR.1.128 Crossbar rotate right immediate ocdets X.ROTR.1.128 Crossbar shift left immediate pecks X.SHL.1.2.O Crossbar shift left immediate signed pecks check overflow X.SHL.1.4.O Crossbar shift left immediate signed pecks check overflow X.SHL.1.4.O Crossbar shift left immediate signed ribbles check overflow X.SHL.1.8.O Crossbar shift left immediate signed ribbles check overflow X.SHL.1.1.6.O Crossbar shift left immediate signed luytes check overflow X.SHL.1.1.6.O Crossbar shift left immediate signed doublets check overflow X.SHL.1.1.6.O Crossbar shift left immediate signed doublets check overflow X.SHL.1.3.2 Crossbar shift left immediate signed doublets check overflow X.SHL.1.3.2 Crossbar shift left immediate signed quadlets X.SHL.1.3.2 Crossbar shift left immediate signed octes check overflow X.SHL.1.3.2 Crossbar shift left immediate signed octes check overflow X.SHL.1.1.2.O Crossbar shift left immediate signed octes check overflow X.SHL.1.1.2.O Crossbar shift left immediate unsigned pecks check overflow X.SHL.1.1.2.O Crossbar shift left immediate unsigned pecks check overflow X.SHL.1.1.2.O Crossbar shift left immediate unsigned pecks check overflow X.SHL.1.1.0.O Crossbar shift left immediate unsigned pecks check overflow X.SHL.1.1.0.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.0.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.1.2.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.1.2.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.1.2.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.1.2.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.1.2.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.1.1.2.O Crossbar shift left immediate unsigned octes check overflow X.SHR.1.1.2 Crossbar shift left immediate unsigned octes check overflow X.SHR.1.1.1.0.O Crossbar shift right immediate unsign	X.ROTR.I.32	Coordy return sold marchine
X.ROTR.1.128 Crossbar shift left immediate pecks X.SHL.1.2 Crossbar shift left immediate pecks check overflow X.SHL.1.4 Crossbar shift left immediate apped pecks check overflow X.SHL.1.4 Crossbar shift left immediate signed nibbles check overflow X.SHL.1.8 Crossbar shift left immediate signed nibbles check overflow X.SHL.1.8 Crossbar shift left immediate signed upres check overflow X.SHL.1.16 Crossbar shift left immediate signed upres check overflow X.SHL.1.16 Crossbar shift left immediate odublets X.SHL.1.16 Crossbar shift left immediate quadlets X.SHL.1.32 Crossbar shift left immediate octies X.SHL.1.32 Crossbar shift left immediate signed odublets check overflow X.SHL.1.64 Crossbar shift left immediate signed octies check overflow X.SHL.1.64 Crossbar shift left immediate signed octies check overflow X.SHL.1.128 Crossbar shift left immediate unsigned nibbles check overflow X.SHL.1.128.0 Crossbar shift left immediate unsigned pecks check overflow X.SHL.1.U.2.0 Crossbar shift left immediate unsigned nibbles check overflow X.SHL.1.U.4.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.4.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned pecks check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHL.1.U.3.0 Crossbar shift left immediate unsigned dipubles check overflow X.SHR.1.0 Crossbar shift left immediate unsigned dipubles X.SHR.1.3 Crossbar signed shift right immediate pecks X.SHR.1.3 Crossbar signed shift right immediate optes X.SHR.1.0 Crossbar shift light immediate unsigned bytes X.SHR.1.0 Crossbar shift light immediate unsigned bytes X.SHR.1.0		
X.S.H.L.1.2.0 Crossbar shift left immediate signed pecks check overflow X.S.H.L.1.4.0 Crossbar shift left immediate signed ribbles check overflow X.S.H.L.1.4.0 Crossbar shift left immediate signed ribbles check overflow X.S.H.L.1.8.0 Crossbar shift left immediate signed trytes check overflow X.S.H.L.1.1.6 Crossbar shift left immediate signed dublets check overflow X.S.H.L.1.1.6 Crossbar shift left immediate signed doublets check overflow X.S.H.L.1.1.6.0 Crossbar shift left immediate ganed doublets check overflow X.S.H.L.1.3.2 Crossbar shift left immediate signed dublets check overflow X.S.H.L.1.3.2.0 Crossbar shift left immediate signed octiets check overflow X.S.H.L.1.3.2.0 Crossbar shift left immediate signed octiets check overflow X.S.H.L.1.2.8 Crossbar shift left immediate signed octiets check overflow X.S.H.L.1.2.8.0 Crossbar shift left immediate signed hexiet check overflow X.S.H.L.1.2.0 Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.1.1.2.0 Crossbar shift left immediate unsigned hibbies check overflow X.S.H.L.1.1.2.0 Crossbar shift left immediate unsigned pixes check overflow X.S.H.L.1.1.3.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.3.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.3.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.3.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.3.0 Crossbar shift left immediate unsigned hexiet check overflow X.S.H.L.1.1.3.1 Crossbar signed shift right immediate pixes X.S.H.R.1.3.2 Crossbar signed shift right immediate octies X.S.H.R.1.3.2 Crossbar signed shift right immediate octies X.S.H.R.1.3.3 Crossbar signed shift right immediate octies X.S.H.R.1.3.4 Crossbar signed shift right immediate octies X.S.H.R.1.3.5 Crossbar signed shift right immediate octies X.S.H.R.1.3.6 Crossbar signed shift right immediate unsigned pixes X.S.H.R.1.3.1 Crossbar signed shift right immediate unsigned octies X.S.H.R.1.3.4 Crossbar shift right immedia		
XSHL1.2.O Crossbar shift left immediate nibbles XSHL1.4.O Crossbar shift left immediate nibbles XSHL1.8.C Crossbar shift left immediate signed nibbles check overflow XSHL1.8.C Crossbar shift left immediate signed uptes check overflow XSHL1.8.O Crossbar shift left immediate signed uptes check overflow XSHL1.16.O Crossbar shift left immediate signed doublets check overflow XSHL1.12.C Crossbar shift left immediate updets XSHL1.32.O Crossbar shift left immediate updets XSHL1.32.O Crossbar shift left immediate updets XSHL1.64.O Crossbar shift left immediate signed quadlets check overflow XSHL1.128.O Crossbar shift left immediate upded expect check overflow XSHL1.128.O Crossbar shift left immediate upded hexlet check overflow XSHL1.128.O Crossbar shift left immediate upded hexlet check overflow XSHL1.128.O Crossbar shift left immediate upded hexlet check overflow XSHL1.128.O Crossbar shift left immediate upded hexlet check overflow XSHL1.128.O Crossbar shift left immediate upded hibbles check overflow XSHL1.128.O Crossbar shift left immediate upded doublets check overflow XSHL1.128.O Crossbar shift left immediate upded doublets check overflow XSHL1.128.O Crossbar shift left immediate upded doublets check overflow XSHL1.128.O Crossbar shift left immediate upded doublets check overflow XSHL1.128.O Crossbar shift left immediate upded doublets check overflow XSHL1.128.O Crossbar shift left immediate upded hexlet check overflow XSHR1.12 Crossbar shift left immediate upded hexlet check overflow XSHR1.12 Crossbar shift left immediate upded hexlet check overflow XSHR1.14 Crossbar shift left immediate upded hexlet check overflow XSHR1.14 Crossbar shift left immediate upded hexlet check overflow XSHR1.14 Crossbar shift left immediate upded hexlet check overflow XSHR1.14 Crossbar shift left immediate upded hexlet check overflow XSHR1.10 Crossbar shift left immediate upded oublets XSHR1.10 Crossbar shift left immediate upded doublets XSHR1.10 Crossbar shift left immedi		
XSHL1.4.O Crossbar shift left immediate signed nibbles check overflow XSHL1.8.O Crossbar shift left immediate bytes XSHL1.8.O Crossbar shift left immediate bytes check overflow XSHL1.1.0.O Crossbar shift left immediate bytes check overflow XSHL1.1.0.O Crossbar shift left immediate doublets check overflow XSHL1.1.0.O Crossbar shift left immediate quadlets check overflow XSHL1.3.2.O Crossbar shift left immediate quadlets check overflow XSHL1.3.2.O Crossbar shift left immediate octies XSHL1.6.O Crossbar shift left immediate signed quadlets check overflow XSHL1.6.O Crossbar shift left immediate signed octiets check overflow XSHL1.1.2.B Crossbar shift left immediate signed hexlet check overflow XSHL1.1.2.O Crossbar shift left immediate unsigned pecks check overflow XSHL1.1.2.O Crossbar shift left immediate unsigned pixes check overflow XSHL1.1.0.O Crossbar shift left immediate unsigned hytes check overflow XSHL1.1.0.O Crossbar shift left immediate unsigned doublets check overflow XSHL1.1.0.O Crossbar shift left immediate unsigned doublets check overflow XSHL1.1.0.O Crossbar shift left immediate unsigned octiets check overflow XSHL1.1.0.O Crossbar shift left immediate unsigned octiets check overflow XSHL1.1.0.O Crossbar shift left immediate unsigned doublets check overflow XSHL1.1.0.1.2.O Crossbar shift left immediate unsigned octiets check overflow XSHL1.1.0.1.2.O Crossbar shift left immediate unsigned doublets check overflow XSHL1.1.0.1.2.O Crossbar shift left immediate unsigned healet check overflow XSHL1.1.0.1.2.O Crossbar shift left immediate unsigned healet check overflow XSHL1.1.0.1.2.O Crossbar shift left immediate unsigned pecks check overflow XSHL1.1.0.1.2.O Crossbar shift left immediate unsigned healet check overflow XSHR1.1.0.1.2. Crossbar signed shift right immediate doublets XSHR1.1.0.0 Crossbar shift right immediate unsigned pecks XSHR1.1.0.1.0 Crossbar shift right immediate unsigned pecks XSHR1.1.0.1.0 Crossbar shift right immediate unsigned doublets XSHR1.1.0.1.0 Crossbar shift right immediate unsign		
X.S.H.L.1.4.O Crossbar shift left immediate bytes X.S.H.L.1.8.C Crossbar shift left immediate bytes X.S.H.L.1.1.6.C Crossbar shift left immediate signed bytes check overflow X.S.H.L.1.1.6.C Crossbar shift left immediate signed bytes check overflow X.S.H.L.1.1.6.C Crossbar shift left immediate doublets X.S.H.L.1.3.2.C Crossbar shift left immediate signed doublets check overflow X.S.H.L.1.3.2.C Crossbar shift left immediate signed quadlets check overflow X.S.H.L.1.3.2.C Crossbar shift left immediate signed octiets check overflow X.S.H.L.1.2.B Crossbar shift left immediate signed hexlet check overflow X.S.H.L.1.2.C Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.1.2.C Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.1.2.C Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.1.2.C Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.2.C Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.6.C Crossbar shift left immediate unsigned octiets check overflow X.S.H.L.1.1.6.C Crossbar shift left immediate unsigned octiets check overflow X.S.H.L.1.1.6.C Crossbar shift left immediate unsigned octiets check overflow X.S.H.L.1.1.1.6.C Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.1.1.1.6.C Crossbar signed shift right immediate pecks X.S.H.R.1.1.6 Crossbar signed shift right immediate doublets X.S.H.R.1.1.6 Crossbar signed shift right immediate doublets X.S.H.R.1.1.6 Crossbar signed shift right immediate doublets X.S.H.R.1.1.1.6 Crossbar signed shift right immediate doublets X.S.H.R.1.1.1.6 Crossbar signed shift right immediate unsigned becks X.S.H.R.1.1.1.8 Crossbar signed shift right immediate unsigned becks X.S.H.R.1.1.1.8 Crossbar signed shift right immediate unsigned becks X.S.H.R.1.1.1.8 Crossbar shift right immediate unsigned bottes X.S.H.R.1.1.1.8 Crossbar shift right immediate unsigned doublets X.S.H.R.1.1.1.6 Crossbar shift right immediate uns		Crossbar shift left immediate signed pecks check overflow
X.S.H.L.1.8. Crossbar shift left immediate bytes X.S.H.L.1.6. Crossbar shift left immediate signed bytes check overflow X.S.H.L.1.6. Crossbar shift left immediate signed doublets check overflow X.S.H.L.1.8.0 Crossbar shift left immediate doublets check overflow X.S.H.L.1.3.2 Crossbar shift left immediate quadlets check overflow X.S.H.L.1.3.2 Crossbar shift left immediate signed quadlets check overflow X.S.H.L.1.3.2 Crossbar shift left immediate signed octets check overflow X.S.H.L.1.2.8 Crossbar shift left immediate signed octets check overflow X.S.H.L.1.1.2.8 Crossbar shift left immediate signed heidet check overflow X.S.H.L.1.1.2.0 Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.1.1.2.0 Crossbar shift left immediate unsigned pixes check overflow X.S.H.L.1.1.4.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.8.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.8.0 Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.1.1.8.0 Crossbar shift left immediate unsigned octets check overflow X.S.H.L.1.1.1.8.0 Crossbar shift left immediate unsigned octets check overflow X.S.H.L.1.1.1.8.0 Crossbar shift left immediate unsigned octets check overflow X.S.H.L.1.1.1.8.0 Crossbar signed shift left immediate unsigned heidet check overflow X.S.H.L.1.1.1.8.0 Crossbar signed shift right immediate pecks X.S.H.R.1.2 Crossbar signed shift right immediate pecks X.S.H.R.1.3.1 Crossbar signed shift right immediate doublets X.S.H.R.1.1.1 Crossbar signed shift right immediate quadlets X.S.H.R.1.1.1 Crossbar signed shift right immediate unsigned bytes X.S.H.R.1.1.1 Crossbar signed shift right immediate unsigned bytes X.S.H.R.1.1.1 Crossbar shift right immediate unsigned bytes X.S.H.R.1.1.1 Crossbar shift right immediate unsigned bytes X.S.H.R.1.1.1 Crossbar shift right immediate unsigned doublets X.S.H.R.1.1.1 Crossbar shift right immediate unsigned doublets X.S.H.R.1.1.1 Crossbar shift right immediate unsigned doublets X.S.H.R.1.1.1 Crossbar shif		
X.S.H.L.I.8.O Crossbar shift left immediate signed bytes check overflow X.S.H.L.I.16.O Crossbar shift left immediate doublets X.S.H.L.I.32.O Crossbar shift left immediate upudlets X.S.H.L.I.32.O Crossbar shift left immediate upudlets X.S.H.L.I.32.O Crossbar shift left immediate upudlets check overflow X.S.H.L.I.32.O Crossbar shift left immediate signed quadlets check overflow X.S.H.L.I.28.O Crossbar shift left immediate signed octets check overflow X.S.H.L.I.128.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.128.O Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.I.I.2.O Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned hytes check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.I.8.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.8 Crossbar signed shift right immediate pecks X.S.H.R.I.8 Crossbar signed shift right immediate unsigned doublets X.S.H.R.I.16 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.18 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.19 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.19 Crossbar shift right immediate unsigned doublets X.S.H.R.I.10 Crossbar shift right immediate unsigned doublets		Crossbar shift left immediate signed nibbles check overflow
XSHILL.16.0 Crossbar shift left immediate doublets XSHIL.13.2 Crossbar shift left immediate upadlets XSHIL.13.2 Crossbar shift left immediate upadlets XSHIL.32.0 Crossbar shift left immediate signed quadlets check overflow XSHIL.16.4 Crossbar shift left immediate signed quadlets check overflow XSHIL.16.6 Crossbar shift left immediate signed octlets check overflow XSHIL.12.8 Crossbar shift left immediate bexlet XSHIL.12.0 Crossbar shift left immediate unsigned hexlet check overflow XSHIL.12.0 Crossbar shift left immediate unsigned pecks check overflow XSHIL.12.0 Crossbar shift left immediate unsigned nibbles check overflow XSHIL.12.0 Crossbar shift left immediate unsigned hytes check overflow XSHIL.10.6 Crossbar shift left immediate unsigned doublets check overflow XSHIL.10.16.0 Crossbar shift left immediate unsigned doublets check overflow XSHIL.10.12.0 Crossbar shift left immediate unsigned octlets check overflow XSHIL.10.12.0 Crossbar shift left immediate unsigned octlets check overflow XSHIL.10.12.0 Crossbar shift left immediate unsigned hexlet check overflow XSHIL.10.12.0 Crossbar shift left immediate unsigned hexlet check overflow XSHIL.10.13.0 Crossbar shift left immediate unsigned hexlet check overflow XSHIL.10.14.0 Crossbar signed shift right immediate pecks XSHR.1.2 Crossbar signed shift right immediate pecks XSHR.1.16 Crossbar signed shift right immediate unsigned pecks XSHR.1.16 Crossbar signed shift right immediate unsigned pecks XSHR.1.12.1 Crossbar signed shift right immediate unsigned pecks XSHR.1.12.1 Crossbar shift right immediate unsigned pecks XSHR.1.14 Crossbar shift right immediate unsigned pecks XSHR.1.16 Crossbar shift right immediate unsigned doublets XSHR.1.18 Crossbar shift right immediate unsigned doublets XSHR.1.19 Crossbar shift right immediate unsigned doublets XSHR.1.10 Crossbar shift right immediate unsigned doublets XSHR.1.11.1 Crossbar shift right immediate unsigned doublets		Crossbar shift left immediate bytes
X.S.H.L.I.1.6.O Crossbar shift left immir liate signed doublets check overflow X.S.H.L.I.32.O Crossbar shift left immediate guadlets X.S.H.L.I.32.O Crossbar shift left immediate signed quadlets check overflow X.S.H.L.I.64.O Crossbar shift left immediate signed octlets check overflow X.S.H.L.I.64.O Crossbar shift left immediate signed octlets check overflow X.S.H.L.I.128.O Crossbar shift left immediate ensigned hexlet check overflow X.S.H.L.I.128.O Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.I.V.2.O Crossbar shift left immediate unsigned pixtes check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.V.3.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.L.I.V.3.O Crossbar signed shift right immediate pecks X.S.H.R.I.4 Crossbar signed shift right immediate bytes X.S.H.R.I.8 Crossbar signed shift right immediate doublets X.S.H.R.I.16 Crossbar signed shift right immediate doublets X.S.H.R.I.128 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.V.4 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.V.4 Crossbar shift right immediate unsigned pecks X.S.H.R.I.V.4 Crossbar shift right immediate unsigned pecks X.S.H.R.I.V.4 Crossbar shift right immediate unsigned doublets X.S.H.R.I.V.16 Crossbar shift right immediate unsigned doublets X.S.H.R.I.V.16 Crossbar shift right immediate unsigned doublets X.S.H.R.I.V.16 Crossbar shift right immediate unsigned doublets		Crossbar shift left immediate signed bytes check overflow
X.SHL.I.32 Crossbar shift left immediate guadlets X.SHL.I.32.O Crossbar shift left immediate signed quadlets check overflow X.SHL.I.64 Crossbar shift left immediate signed octlets check overflow X.SHL.I.128 Crossbar shift left immediate signed octlets check overflow X.SHL.I.128.O Crossbar shift left immediate signed hexlet check overflow X.SHL.I.128.O Crossbar shift left immediate unsigned pecks check overflow X.SHL.I.U.2.O Crossbar shift left immediate unsigned nibbles check overflow X.SHL.I.U.4.O Crossbar shift left immediate unsigned nibbles check overflow X.SHL.I.U.8.O Crossbar shift left immediate unsigned doublets check overflow X.SHL.I.U.16.O Crossbar shift left immediate unsigned quadlets check overflow X.SHL.I.U.128.O Crossbar shift left immediate unsigned octlets check overflow X.SHL.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.SHL.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.SHR.I.2 Crossbar signed shift right immediate pecks X.SHR.I.4 Crossbar signed shift right immediate pecks X.SHR.I.8 Crossbar signed shift right immediate doublets X.SHR.I.16 Crossbar signed shift right immediate doublets X.SHR.I.18 Crossbar signed shift right immediate octlets X.SHR.I.19 Crossbar signed shift right immediate unsigned pecks X.SHR.I.10 Crossbar shift right immediate unsigned pecks X.SHR.I.118 Crossbar shift right immediate unsigned pecks X.SHR.I.108 Crossbar shift right immediate unsigned hexlet X.SHR.I.109 Crossbar shift right immediate unsigned hexlet X.SHR.I.109 Crossbar shift right immediate unsigned hexlet X.SHR.I.109 Crossbar shift right immediate unsigned doublets		
X.S.H.L.1.32.O Crossbar shift left immediate signed quadlets check overflow X.S.H.L.1.64.O Crossbar shift left immediate signed octlets check overflow X.S.H.L.1.1.28 Crossbar shift left immediate signed hexiet check overflow X.S.H.L.1.1.28.O Crossbar shift left immediate signed hexiet check overflow X.S.H.L.1.1.28.O Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.1.1.2.O Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.1.1.2.O Crossbar shift left immediate unsigned hytes check overflow X.S.H.L.1.1.0.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.0.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.0.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.1.0.O Crossbar shift left immediate unsigned hexiet check overflow X.S.H.L.1.1.0.O Crossbar shift left immediate unsigned hexiet check overflow X.S.H.L.1.1.0.O Crossbar shift left immediate unsigned hexiet check overflow X.S.H.L.1.1.0.O Crossbar signed shift right immediate pecks X.S.H.R.1.2 Crossbar signed shift right immediate pecks X.S.H.R.1.2 Crossbar signed shift right immediate bytes X.S.H.R.1.1.0 Crossbar signed shift right immediate doublets X.S.H.R.1.1.0 Crossbar signed shift right immediate octlets X.S.H.R.1.1.2 Crossbar signed shift right immediate octlets X.S.H.R.1.1.2 Crossbar signed shift right immediate unsigned pecks X.S.H.R.1.1.0 Crossbar shift right immediate unsigned nibbles X.S.H.R.1.1.0 Crossbar shift right immediate unsigned hexiet X.S.H.R.1.1.0 Crossbar shift right immediate unsigned doublets X.S.H.R.1.1.0.0 Crossbar shift right immediate unsigned doublets X.S.H.R.1.1.0.0 Crossbar shift right immediate unsigned doub		Crossbar shift left immr liate signed doublets check overflow
X.SHL.I.64. Crossbar shift left immediate ordets X.SHL.I.28. Crossbar shift left immediate signed ordets check overflow X.SHL.I.128. Crossbar shift left immediate signed hexlet check overflow X.SHL.I.128.O. Crossbar shift left immediate unsigned hexlet check overflow X.SHL.I.U.2.O. Crossbar shift left immediate unsigned pecks check overflow X.SHL.I.U.4.O. Crossbar shift left immediate unsigned highes check overflow X.SHL.I.U.8.O. Crossbar shift left immediate unsigned bytes check overflow X.SHL.I.U.16.O. Crossbar shift left immediate unsigned doublets check overflow X.SHL.I.U.12.O. Crossbar shift left immediate unsigned quadlets check overflow X.SHL.I.U.32.O. Crossbar shift left immediate unsigned ocdets check overflow X.SHL.I.U.128.O. Crossbar shift left immediate unsigned hexlet check overflow X.SHL.I.U.128.O. Crossbar shift left immediate unsigned hexlet check overflow X.SHR.I.2. Crossbar signed shift right immediate pecks X.SHR.I.4. Crossbar signed shift right immediate bytes X.SHR.I.8. Crossbar signed shift right immediate doublets X.SHR.I.16. Crossbar signed shift right immediate doublets X.SHR.I.128. Crossbar signed shift right immediate ottelets X.SHR.I.128. Crossbar signed shift right immediate ottelets X.SHR.I.128. Crossbar shift right immediate unsigned pecks X.SHR.I.U.2. Crossbar shift right immediate unsigned pocks X.SHR.I.U.4. Crossbar shift right immediate unsigned doublets X.SHR.I.U.8. Crossbar shift right immediate unsigned doublets X.SHR.I.U.16. Crossbar shift right immediate unsigned doublets		
X.S.H.L.1.28 Crossbar shift left immediate signed octlets check overflow X.S.H.L.1.128.0 Crossbar shift left immediate signed hexiet check overflow X.S.H.L.1.28.0 Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.1.2.0 Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.1.2.0 Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.1.2.0 Crossbar shift left immediate unsigned hytes check overflow X.S.H.L.1.2.0 Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.1.2.0 Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.1.2.128.0 Crossbar shift left immediate unsigned octlets check overflow X.S.H.L.1.2.128.0 Crossbar signed shift right immediate pecks X.S.H.R.1.2 Crossbar signed shift right immediate pecks X.S.H.R.1.8 Crossbar signed shift right immediate bytes X.S.H.R.1.16 Crossbar signed shift right immediate doublets X.S.H.R.1.16 Crossbar signed shift right immediate doublets X.S.H.R.1.128 Crossbar signed shift right immediate octlets X.S.H.R.1.128 Crossbar signed shift right immediate octlets X.S.H.R.1.128 Crossbar signed shift right immediate octlets X.S.H.R.1.128 Crossbar shift right immediate unsigned pecks X.S.H.R.1.128 Crossbar shift right immediate unsigned hytes X.S.H.R.1.128 Crossbar shift right immediate unsigned hytes X.S.H.R.1.128 Crossbar shift right immediate unsigned doublets X.S.H.R.1.132 Crossbar shift right immediate unsigned doublets		Crossbar shift left immediate signed quadlets check overflow
X.S.H.L.I.128 Crossbar shift left immediate hexiet X.S.H.L.I.128.O Crossbar shift left immediate signed hexiet check overflow X.S.H.L.I.U.2.O Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.I.U.4.O Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.I.U.8.O Crossbar shift left immediate unsigned hytes check overflow X.S.H.L.I.U.16.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.U.16.O Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.I.U.128.O Crossbar shift left immediate unsigned octiets check overflow X.S.H.L.I.I.128.O Crossbar shift left immediate unsigned hexiet check overflow X.S.H.L.I.I.128.O Crossbar signed shift right immediate pecks X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.16 Crossbar signed shift right immediate doublets X.S.H.R.I.128 Crossbar signed shift right immediate doublets X.S.H.R.I.128 Crossbar signed shift right immediate octiets X.S.H.R.I.128 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.128 Crossbar shift right immediate unsigned pecks X.S.H.R.I.128 Crossbar shift right immediate unsigned bytes X.S.H.R.I.128 Crossbar shift right immediate unsigned doublets X.S.H.R.I.128 Crossbar shift right immediate unsigned bytes X.S.H.R.I.149 Crossbar shift right immediate unsigned doublets		
X.S.H.L.I.128.O Crossbar shift left immediate signed hexiet check overflow X.S.H.L.I.U.2.O Crossbar shift left immediate unsigned pecks check overflow X.S.H.L.I.U.4.O Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.I.U.8.O Crossbar shift left immediate unsigned bytes check overflow X.S.H.L.I.U.16.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.U.32.O Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.I.U.128.O Crossbar shift left immediate unsigned octiets check overflow X.S.H.L.I.U.128.O Crossbar shift left immediate unsigned hexiet check overflow X.S.H.R.I.2 Crossbar signed shift right immediate pecks X.S.H.R.I.4 Crossbar signed shift right immediate pibbles X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.8 Crossbar signed shift right immediate doublets X.S.H.R.I.12 Crossbar signed shift right immediate quadlets X.S.H.R.I.128 Crossbar signed shift right immediate octiets X.S.H.R.I.128 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.128 Crossbar shift right immediate unsigned pecks X.S.H.R.I.U.14 Crossbar shift right immediate unsigned bytes X.S.H.R.I.U.14 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.18 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.18 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.19 Crossbar shift right immediate unsigned doublets		Crossbar shift left immediate signed octlets check overflow
X.S.H.L.I.U.4.O X.S.H.L.I.U.4.O X.S.H.L.I.U.8.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.U.8.O X.S.H.L.I.U.8.O Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.I.U.8.O X.S.H.L.I.U.8.O X.S.H.R.I.2 Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.2 X.S.H.R.I.8 Crossbar signed shift right immediate pecks X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.8 X.S.H.R.I.8 Crossbar signed shift right immediate doublets X.S.H.R.I.16 Crossbar signed shift right immediate quadlets X.S.H.R.I.128 Crossbar signed shift right immediate octiets X.S.H.R.I.128 Crossbar signed shift right immediate nexlet X.S.H.R.I.128 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.U.1 X.S.H.R.I.U.1 Crossbar shift right immediate unsigned nibbles X.S.H.R.I.U.1 X.S.H.R.I.U.1 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.1		
X.S.H.L.I.U.4.O Crossbar shift left immediate unsigned nibbles check overflow X.S.H.L.I.U.8.O Crossbar shift left immediate unsigned bytes check overflow X.S.H.L.I.U.16.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.U.32.O Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.I.U.64.O Crossbar shift left immediate unsigned octlets check overflow X.S.H.I.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.2 Crossbar signed shift right immediate pecks X.S.H.R.I.4 Crossbar signed shift right immediate nibbles X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.8 Crossbar signed shift right immediate doublets X.S.H.R.I.16 Crossbar signed shift right immediate quadlets X.S.H.R.I.128 Crossbar signed shift right immediate octlets X.S.H.R.I.128 Crossbar signed shift right immediate hexlet X.S.H.R.I.128 Crossbar shift right immediate unsigned pecks X.S.H.R.I.U.14 Crossbar shift right immediate unsigned bytes X.S.H.R.I.U.16 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.16 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate signed hexlet check overflow
X.S.H.L.I.U.8.O Crossbar shift left immediate unsigned hytes check overflow X.S.H.L.I.U.16.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.U.32.O Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.I.U.64.O Crossbar shift left immediate unsigned octiets check overflow X.S.H.L.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.2 Crossbar signed shift right immediate pecks X.S.H.R.I.4 Crossbar signed shift right immediate nibbles X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.16 Crossbar signed shift right immediate doublets X.S.H.R.I.12 Crossbar signed shift right immediate quadlets X.S.H.R.I.128 Crossbar signed shift right immediate octlets X.S.H.R.I.128 Crossbar signed shift right immediate hexlet X.S.H.R.I.U.2 Crossbar shift right immediate unsigned pecks X.S.H.R.I.U.14 Crossbar shift right immediate unsigned nibbles X.S.H.R.I.U.18 Crossbar shift right immediate unsigned bytes X.S.H.R.I.U.18 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.19 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate unsigned pecks check overflow
X.S.H.L.I.U.16.O Crossbar shift left immediate unsigned doublets check overflow X.S.H.L.I.U.32.O Crossbar shift left immediate unsigned quadlets check overflow X.S.H.L.I.U.64.O Crossbar shift left immediate unsigned octlets check overflow X.S.H.L.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.2 Crossbar signed shift right immediate pecks X.S.H.R.I.4 Crossbar signed shift right immediate nibbles X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.16 Crossbar signed shift right immediate doublets X.S.H.R.I.32 Crossbar signed shift right immediate quadlets X.S.H.R.I.32 Crossbar signed shift right immediate octlets X.S.H.R.I.128 Crossbar signed shift right immediate hexlet X.S.H.R.I.128 Crossbar shift right immediate unsigned pecks X.S.H.R.I.1.U.2 Crossbar shift right immediate unsigned nibbles X.S.H.R.I.U.4 Crossbar shift right immediate unsigned bytes X.S.H.R.I.U.8 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.16 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.16 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.U.104 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.105 Crossbar shift right immediate unsigned octlets X.S.H.R.I.U.106 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate unsigned nibbles check overflow
XSHL.I.U.32.O Crossbar shift left immediate unsigned quadlets check overflow X.SHL.I.U.64.O Crossbar shift left immediate unsigned octets check overflow X.SHL.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.SHR.I.2 Crossbar signed shift right immediate pecks X.SHR.I.4 Crossbar signed shift right immediate nibbles X.SHR.I.8 Crossbar signed shift right immediate bytes X.SHR.I.16 Crossbar signed shift right immediate doublets X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.32 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.128 Crossbar shift right immediate unsigned pecks X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned bytes X.SHR.I.U.8 Crossbar shift right immediate unsigned doublets X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.32 Crossbar shift right immediate unsigned octlets X.SHR.I.U.34 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate unsigned bytes check overflow
X.S.H.I.I.U.128.O Crossbar shift left immediate unsigned octlets check overflow X.S.H.I.I.U.128.O Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.2 Crossbar signed shift right immediate pecks X.S.H.R.I.4 Crossbar signed shift right immediate nibbles X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.16 Crossbar signed shift right immediate doublets X.S.H.R.I.12 Crossbar signed shift right immediate quadlets X.S.H.R.I.128 Crossbar signed shift right immediate octlets X.S.H.R.I.128 Crossbar signed shift right immediate hexlet X.S.H.R.I.U.2 Crossbar shift right immediate unsigned pecks X.S.H.R.I.U.4 Crossbar shift right immediate unsigned nibbles X.S.H.R.I.U.18 Crossbar shift right immediate unsigned bytes X.S.H.R.I.U.18 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.18 Crossbar shift right immediate unsigned doublets X.S.H.R.I.U.18 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.U.194 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.U.195 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.U.196 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate unsigned doublets check overflow
X.S.H.R.I.2 Crossbar shift left immediate unsigned hexlet check overflow X.S.H.R.I.2 Crossbar signed shift right immediate pecks X.S.H.R.I.8 Crossbar signed shift right immediate bytes X.S.H.R.I.16 Crossbar signed shift right immediate doublets X.S.H.R.I.17 Crossbar signed shift right immediate doublets X.S.H.R.I.18 Crossbar signed shift right immediate quadlets X.S.H.R.I.19 Crossbar signed shift right immediate octlets X.S.H.R.I.19 Crossbar signed shift right immediate octlets X.S.H.R.I.19 Crossbar signed shift right immediate unsigned pecks X.S.H.R.I.19 Crossbar shift right immediate unsigned nibbles X.S.H.R.I.19 Crossbar shift right immediate unsigned bytes X.S.H.R.I.19 Crossbar shift right immediate unsigned doublets X.S.H.R.I.19 Crossbar shift right immediate unsigned doublets X.S.H.R.I.19 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.19 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.19 Crossbar shift right immediate unsigned quadlets X.S.H.R.I.19 Crossbar shift right immediate unsigned octlets X.S.H.R.I.19 Crossbar shift right immediate unsigned octlets		Crossoar shift left immediate unsigned quadlets check overflow
X.SHR.I.2 Crossbar signed shift right immediate pecks X.SHR.I.8 Crossbar signed shift right immediate bytes X.SHR.I.16 Crossbar signed shift right immediate doublets X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.64 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.34 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.35 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate unsigned octlets check overflow
X.SHR.I.4 Crossbar signed shift right immediate nibbles X.SHR.I.8 Crossbar signed shift right immediate bytes X.SHR.I.16 Crossbar signed shift right immediate doublets X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.64 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.34 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.36 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.34 Crossbar shift right immediate unsigned octlets		Crossbar shift left immediate unsigned hexlet check overflow
X.SHR.I.8 Crossbar signed shift right immediate bytes X.SHR.I.16 Crossbar signed shift right immediate doublets X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.64 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.34 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.35 Crossbar shift right immediate unsigned octlets		
X.SHR.I.16 Crossbar signed shift right immediate doublets X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.64 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned quadlets		
X.SHR.I.32 Crossbar signed shift right immediate quadlets X.SHR.I.64 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.34 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.35 Crossbar shift right immediate unsigned octlets		
X.SHR.I.64 Crossbar signed shift right immediate octlets X.SHR.I.128 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		
X.SHR.I.U.2 Crossbar signed shift right immediate hexlet X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		Crossbar signed shift right immediate quadlets
X.SHR.I.U.2 Crossbar shift right immediate unsigned pecks X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		
X.SHR.I.U.4 Crossbar shift right immediate unsigned nibbles X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		Crossbar signed shift right immediate hexlet
X.SHR.I.U.8 Crossbar shift right immediate unsigned bytes X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		Crossbar shift right immediate unsigned pecks
X.SHR.I.U.16 Crossbar shift right immediate unsigned doublets X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		Crossbar shift right immediate unsigned nibbles
X.SHR.I.U.32 Crossbar shift right immediate unsigned quadlets X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		
X.SHR.I.U.64 Crossbar shift right immediate unsigned octlets		
X.SHR.I.U.128 Crossbar shift right immediate unsigned hexlet		Crossbar shift right immediate unsigned octlets
	X.SHR.J.U.128	Crossbar shift right immediate unsigned hexlet

Equivalencies

X.COPY	Crossbar copy
XNOP	Crossbar no operation

V.CO.		
X.COPY rd=rc	←	X.ROTL.1.128 rd=rc.0
XNOP		X.COPY r0=r0

Redundancies

X.ROTLI.gsize rd=rc,0	A X.COPY rd=rc
X.ROTR.I.gsize rd=rc.0	A XCOPY rd=rc
X.ROTR.I.gsize rd= c.shift	A XROTLI.gsize rd=rc.gsize-shift
X.SHL.Lgsize rd=rc,0	A XCOPY rd=rc
XSHLI.gsize.O rd=rc,0	A XCOPY rd=rc
XSHLI.U.gsize.O rd=rc,0	A XCOPY rd=rc
X.SHR.I.gsize rd=rc,0	A XCOPY re-re
XSHR.I.U.gsize rd=rc,0	⇔ XCOPY rd=rc

Selection

class	ор		size		
precision	COMPRESS.I EXPAND.I	COMPRESS.I.U EXPAND.I.U	2 4.8 16	32 64	128
shift	I		2 4 8 16	32 64	128
сору	COPY				

Format

X.op.size rd=rc,shift

rd=xopsize(rc,shift)

31	24 23	18	17	12 11	6	5 n
XSHIFTI		rd	rc		simm	OD
8		6	6		<u> </u>	

t ← 256-2*size+shift

 $op_{1.0} \leftarrow t_{7.6}$

simm ← t_{5 0}

Description

A 128-bit value is taken from the contents of register rc. The second operand is taken from simm. The specified operation is performed, and the result is placed in register rd.

Definition

def CrossbarShortImmediate(op,rd,rc,simm)

case (op1.0 11 simm) of

0..127:

size ← 128

128..191:

size ← 64

```
192.223:
           size ← 32
      224..239:
           size :- 16
     240. 247:
           size ← 8
     248..251:
           size ← 4
     252..253:
          size \leftarrow 2
     254..255:
          raise ReservedInstruction
endcase
shift ← (opo 11 simm) and (size-1)
c ← RegReadirc, 128
case lops 2 11 02) of
     X.COMPRESS.I.
          hsize ← size/2
          for i ← 0 to 64-hsize by hsize
               if shift ≤ hsize then
                     alehsize_1 i ← Ciereshift+hsize-1 ileieshift
                     antisze-1 : ← Cshift-huze 11 Cinnusze-1 ininshift
               endif
          endfor
         a127 64 ← 0
    X COMPRESS.I U:
         hsize - size/2
         for i - 0 to 64-hsize by hsize
               if shift ≤ hsize then
                    aleurise-1:1 - Cienstriftensize-1:1elestrift
                    amphsize-1 ( ← Oshift-hsize 11 Ci+i+size-1 i+i+shift
              endif
         endfor
         2127 64 ← 0
   X EXPANDI-
         hsize - size/2
         for i 

0 to 64-hsize by hsize
              if shift ≤ hsize then
                   Sheepsel in this charge of 11 Cleuster 1 11 Oshift
              else
                    globelise-1 for ← Chelise-spite-1 ! 11 Ospite
              er dif
        endfor
   X EXPANDIU
        hsize - size/2
        for 1 - 0 to 64-hsize by hsize
              if shift & hsize then
                   gloiotisc-1 iol e Outisc-strift 11 Clotitisco-1 i 11 Ostrift
              cise
                   Properties - Construction 1 1 Ostate
```

```
endif
            endfor
       X.SHLI:
            for i \leftarrow 0 to 128-size by size
                  dispre-1.1 ← Coste-1-shift.11 Oshift
            endfor
      X.SHLI.O:
            for i \leftarrow 0 to 128-size by size
                  # Chaze-1...insze-1-shift # chaze-1-shift then
                       raise FixedPointAnthmetic
                  division 1.1 ← Character 1-shift 11 1 Oshift
            endfor
      X.SHL.I.U.O:
            for i ← 0 to 128-size by size
                 if Cosize-1, osize-shift # Oshift then
                       raise FixedPointArithmetic
                 Provided 1.1 ← Character 1-strict, 11 Ostrict
           endfor
      X ROTRA
           for i ← 0 to 128-size by size
                 ansize-1... ← Coshift-1... 11 Cosize-1. oshift
           endfor
     X.SHR.I:
           for i ← 0 to 128-size by size
                amsize-1 : ← :shift | 11 Cmsize-1..mshift
           endfor
     X.SHR.I.U:
           for i ← 0 to 128-size by size
                Shester1.1 ← Oshift 11 Chester1.Heshift
           endfor
endcase
RegWritefrd, 128, a)
```

Exceptions

enddef

Fixed point anthmetic Peserved Instruction

Crossbar Short Immediate Inplace

These operations take operands from two registers and a short immediate value, perform operations on partitions of bits in the operands, and place the concatenated results in the second register.

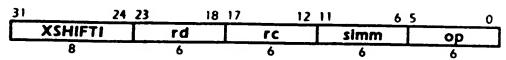
Opt. ation codes

Crossbar shift left merge immediate pecks
Crossbar shift left merge immediate nibbles
Crossbar shift left merge immediate bytes
Crossbar shift left merge immediate doublets
Crossbar shift left merge immediate quadlets
Crossbar shift left merge immediate octlets
Crossbar shift left merge immediate hexlet
Crossbar shift right merge immediate pecks
Crossbar shift right merge immediate nibbles
Crossbar shift right merge immediate bytes
Crossbar shift right merge immediate doublets
Crossbar shift right merge immediate quadlets
Crossbar shift right merge immediate octlets
Crossbar shift right merge immediate hexlet

Format

X.op.size rd@rc,shift

rd=xopsize(rc,shift)



t ← 256-2*size+shift

 $op_{1.0} \leftarrow t_{7.6}$

simm ← t_{5.0}

Description

Two 128-bit values are taken from the contents of registers rd and rc. A third operand is taken from simm. The specified operation is performed, and the result is placed in register rd.

This instruction is undefined and causes a reserved instruction exception if the simm field is greater or equal to the size specified.

Definition

```
def CrossbarShortImmediateInplace[op,rd,rc,simm]
      case top1.0 11 simml of
            0..127:
                 size ← 128
            128.191:
                size ← 64
            192.223:
                sze ← 32
           224..239:
                size ← 16
           240..247:
                size ← 8
           248..251:
                size ← 4
          252..253:
                size \leftarrow 2
           254..255:
                raise ReservedInstruction
      endcase
     shift ← lopo 11 simml and (size-1)
     C ← RegReadfrc, 128)
     d ← RegReadird, 1281
     for i \leftarrow 0 to 128-size by size
          case (op5.2 11 02) of
               X.SHR.M.I:
                     Prospert : ← Coshift-1.1 | | diosze-1.ioshift
               X.SHL.M.I:
                    dissize 1 1 ← dissize 1-shift. 1 1 Cushift-1...
          endcase
     endfor
     RegWntefrd, 128, at
enddef
```

Exceptions

Reserved Instruction

Crossbar Shuffle

These operations take operands from two registers, perform operations on partitions of bits in the operands and place the concatenated results in a register.

Operation codes

X.SHUFFLE.4	Crossbar shuffle within pecks
X.SHUFFLE.8	Crossbar shuffle within bytes
X.SHUFFLE.16	Crossbar shuffle within doublets
X.SHUFFLF.32	Crossbar shuffle within quadlets
X.SHUFFLE.64	Crossbar shuffle within octlets
X.SHUFFLE.128	Crossbar shuffle within hexlet
X.SHUFFLE.256	Crossbar shuffle within triclet

Format

X.SHUFFLE.256 rd=rc,rn,v,w,h X.SHUFFLE.size rd=rcb,v,w

rd=xshuffle256(rc,rb,v,w,h) rd=xshufflesize(rcb,v,w)

31	24	23	18	17	12	11	6 5		0
X.SHU	IFFLE	r	đ	r	C	ri	,	ор	
8		6				6		6	_

 $rc \leftarrow rb \leftarrow rcb$

x←log2(size)

y←log2M

z-logz(w)

op $\leftarrow (|x^*x^*x-3^*x^*x-4^*x|/6-|z^*z-z|/2+x^*z+y| + |size=256|*(h^*32-56)|$

Description

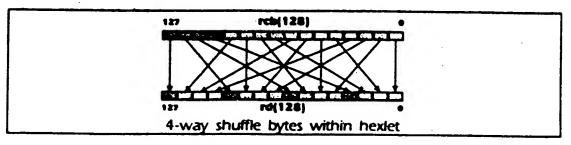
One of two operations are performed depending on whether the re and rb fields are equal.

If the rc and rb fields are equal, a 128-bit operand is taken from the contents of register rc. Items of size v are divided into w piles and shuffled together, within groups of size bits, according to the value of op. The result is placed in register rd.

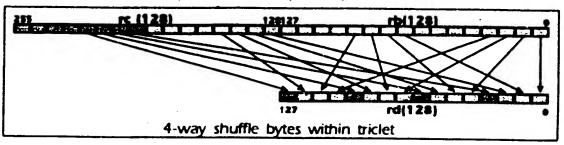
If the re and rb fields are not equal, the contents of registers re and rb are catenated into a 256-bit operand. Items of size v are divided into w piles and shuffled together, according to the value of op. Depending on the value of h, a sub-field of op, the low 128 bits (h=0), or the high 128 bits (h=1) of the 256-bit shuffled contents are selected as the result. The result is placed in register rd.

This instruction is undefined and causes a reserved instruction exception if rc and rb are not equal and the op field is greater or equal to 56, or if rc and rb are equal and op4..0 is greater or equal to 28.

A crossbar 4-way shuffle of bytes within hexlet instruction (X.SHUFFLE.128 rd=rcb,8,4) divides the 128-bit operand into 16 bytes and partitions the bytes 4 ways (indicated by varying shade in the diagram below). The 4 partitions are perfectly shuffled, producing a 128-bit result.



A crossbar 4-way shuffle of bytes within triclet instruction (X.SHUFFLE.256 rd=rc,rb,8,4,0) catenates the contents of rc and rb, then divides the 256-bit content into 32 bytes and partitions the bytes 4 ways (indicated by varying shade in the diagram below). The low-order halves of the 4 partitions are perfectly shuffled, producing a 128-bit result.



Instruction Set Crossbar Shuffle

Changing the last immediate value h to 1 (X.SHUFFI.E.256 rd=rc,rb,8,4,1) modifies the operation to perform the same function on the high-order halves of the 4 partitions.

When re and rb are equal, the table below shows the value of the op field and associated values for size, v, and w.

QD	size	٧	w
ор 0	4	1	2
1	8	1	2
2	8	2	2
3	8	1	4
4	16	1	2
5	16	2	2
6	16	2 4	2
7	16	1	4
1 2 3 4 5 6 7 8	16	2	2 2 4 2 2 2 4 4
9	- 16	1	8
10	32	1 2 1 1	2 2 2 2 4
11	32	2	2
12	32 32	4	2
13	32	4 8	. 2
14	32	1	4
15	32 32	2	4
16	32 32	4	4
17	32	1	8
18	32	2	8
19	32 32	4 1 2 1 1	16
20	64	1	2
21	64	2	2
22	64	4	2
22 23	64	8	2
24	. 64	16	2 2 2 2 2 4 4
24 25	64	1	4
26	64	2 4	4
27	64	4	4

		_	
ор	size	٧	w
28	64	8	4
29	64	1	8
30	64	2	8
31	64	4	8
32	64	1	16
33	64	2	16
34	64	1	32
35	128	1	2
36	128	2	
37	128	4	2
38	128	8	2
39	128	16	2
40	128	32	2
41	128	1	2 4
42	128	2	4
43	128	4	4
44	128	8	4
45	128	16	4
46	128	1	8
47	128	2	8
48	128	4	8
49	128	8	8
50	128	1	16
51	128	2	16
52	128	4	16
53	128	i	32
54	128	ż	32
55	128	1	64
ננו	120	•	U .

When re and rb are not equal, the table below shows the value of the op4.0 field and associated values for size, v, and w: Op5 is the value of h, which controls whether the low-order or high-order half of each partition is shuffled into the result.

OP40	size	٧	W.
0	256	1	2
1	256	2	2
2	256	4	2
3	256	8	2
4	256	16	2
5	256	32	2 2 2 2 2
6	256	64	2
7	256	1	4
8	256	• 2	4
9	256	4	4
10	256	8	4
11.	256	16	4
12	256	32	4
13	256	1	8
14	256	2	8
15	256	4	8
16	256	8	8
17	256	16	8
18	256	1	16
- 19	256	2	16
20	256	4	16
21	256	8	16
22	256	1	32
23	256	2	32
24	256	4	32
25	256	1	64
26	256	2	64
27	256	11	128

Definition

```
def CrossbarShuffle(major,rd,rc,rb,op)

c ← RegRead(rc, 128)
b ← RegRead(rb, 128)
if rc=rb then
casc op of
0..55:

for x ← 2 to 7; for y ← 0 to x-2; for z ← 1 to x-y-1
if op = ((x*x*x-3*x*x-4*x)/6-(z*z-z)/2+x*z-y) then
for i ← 0 to 127

a<sub>i</sub> ← C(i<sub>6,x</sub> | 1 | i<sub>y+z-1</sub> y | 1 | i<sub>x-1,y+z</sub> | 1 | i<sub>y-1</sub> o)
end
endif
endior, endior; endior
56..63:
raise Reservedinstruction
```

Reserved Instruction

```
endcase
     elseif
           case op4_0 of
                0..27:
                      cb ← c II b
                      8 \rightarrow x
                      h ← ops
                      for y \leftarrow 0 to x-2; for z \leftarrow 1 to x-y-1
                            if op_{4.0} = ((17^*z-z^*z)/2-8+y) then
                                 for i ← h*128 to 127+h*128
                                       ai-nº128 - Coffyez-1,y 11 ix-1,yez 11 iy-1.0
                            endif
                      endfor; endfor
                28_31:
                      raise ReservedInstruction
           endcase
     endif
     RegWrite(rd, 128, a)
ಆಸರಣೆ
Exceptions
```

MicroUnity

Crossbar Swizzle

These operations perform calculations with a general register value and immediate values, placing the result in a general register.

Operation codes

XSWIZZLE	Cearchar pressla	
I V 3 M I K K L E	Crossbar swizzle	

Format

X.SWIZZLE rd=rc,icopy,iswap

rd=xswizzle(rc,icopy,iswap)

31	26	2524 23		18 17	12 11	6	5 0
X.SWI	ZZLE	Ih	rd	r	C	Icopya	Iswapa
6		2	6	6		5	6

```
icopya \leftarrow icopy<sub>5.0</sub> iswapa \leftarrow iswap<sub>5.0</sub> ih \leftarrow icopy<sub>6.11</sub> iswap<sub>6</sub>
```

Description

The contents of register re are fetched, and 7-bit immediate values, icopy and iswap, are constructed from the 2-bit ih field and from the 6-bit icopya and iswapa fields. The specified operation is performed on these operands. The result is placed into register rd.

Definition

```
def GroupSwizzleimmediate(ih,rd,rc,icopya,iswapa) as icopy ← ih1 11 icopya iswap ← ih0 11 iswapa c ← RegRead(rc, 128) for i ← 0 to 127 a ← C(i & icopy) iswap endfor RegWrite(rd, 128, a) enddef
```

Exceptions none

Crossbar Ternary

These operations take three values from registers, perform a group of calculations on partitions of bits of the operands and place the catenated results in a fourth register.

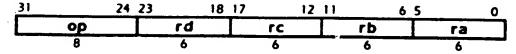
Operation codes

X.SELECT.8	Crossbar select bytes

Format

op ra=rd,rc,rb

ra=op(rd,rc,rb)



Description

The contents of registers rd, rc, and rb are fetched. The specified operation is performed on these operands. The result is placed into register ra.

Definition

```
def CrossbarTernary(op.rd,rc,rb,ra) as

d ← RegRead(rd, 128)

c ← RegRead(rc, 128)

b ← RegRead(rb, 128)

dc ← d | | | c

for | ← 0 to 15

J ← b8°μ4, 8°1

a8°μ7,8°1 ← dC8°μ7,8°3

endfor

RegWrite(ra, 128, a)

enddef
```

Exceptions

none

Ensemble

These operations take operands from two registers, perform operations on partitions of bits in the operands, and place the concatenated results in a third register.

Operation codes

E.CON.8	Ensemble convolve signed bytes
1.CON.16	Ensemble convolve signed doublets
E.CON.32	Ensemble convolve signed quadlets
E.CON.64	Ensemble convolve signed octlets
E.CON.C.8	Ensemble convolve complex bytes
E.CON.C.16	Ensemble convolve complex doublets
E.CON.C.32	Ensemble convolve complex quadlets
E.CON.M.8	Ensemble convolve mixed-signed bytes
E.CON.M.16	Ensemble convolve mixed-signed doublets
E.CON.M.32	Ensemble convolve mixed-signed quadlets
E.CON.M.64	Ensemble convolve mixed-signed octlets
E.CON.U.8	Ensemble convolve unsigned bytes
E.CON.U.16	Ensemble convolve unsigned doublets
E.CON.U.32	Ensemble convolve unsigned quadlets
E.CON.U.64	Ensemble convolve unsigned octlets
E.DIV.64	Ensemble divide signed octlets
E.DIV.U.64	Ensemble divide unsigned octlets
E.MUL.8	Ensemble multiply signed bytes
E.MUL.16	Ensemble multiply signed doublets
E.MUL.32	Ensemble multiply signed quadlets
E.MUL.64	Ensemble multiply signed octlets
E.MULSUM.8	Ensemble multiply sum signed bytes
E.MULSUM.16	Ensemble multiply sum signed doublets
E.MULSUM.32	Ensemble multiply sum signed quadlets
E.MUL.SUM.64	Ensemble multiply sum signed octlets
E.MUL.C.8	Ensemble complex multiply bytes
EMUL.C.16	Ensemble complex multiply doublets
E.MUL.C.32	Ensemble complex multiply quadleu
E.MUL.M.8	Ensemble multiply mixed-signed bytes
E.MUL.M.16	Ensemble multiply mixed-signed doublets
E.MUL.M.32	Ensemble multiply mixed-signed quadlets
E.MUL.M.64	Ensemble multiply mixed-signed octlets
E.MUL.P.8	Ensemble multiply polynomial bytes
E.MUL.P.16	Ensemble multiply polynomial doublets
E.MUL.P.32	Ensemble multiply polynomial quadlets
E.MUL.P.64	Ensemble multiply polynomial octlets
E.MUL.SUM.C.8	Ensemble multiply sum complex bytes
E.MUL.SUM.C.16	Ensemble multiply sum complex doublets
E.MUL.SUM.C.32	Ensemble multiply sum complex quadlets

E.MUL.SUM.M.8	Ensemble multiply sum mixed-signed bytes
E.MUL.SUM.M.16	Ensemble multiply sum mixed-signed doublets
E.MULSUM.M.32	Ensemble multiply sum mixed-signed quadlets
E.MULSUM.M.64	Ensemble multiply sum mixed-signed octlets
E.MULSUM.U.8	Ensemble multiply sum unsigned bytes
E.MULSUM.U.16	Ensemble multiply sum unsigned doublets
E.MULSUM.U.32	Ensemble multiply sum unsigned quadlets
E.MULSUM.U.64	Ensemble multiply sum unsigned octlets
E.MUL.U.8	Ensemble multiply unsigned bytes
E.MULU.16	Ensemble multiply unsigned doublets
E.MUL.U.32	Ensemble multiply unsigned quadlets
E.MULU.64	Ensemble multiply unsigned octlets

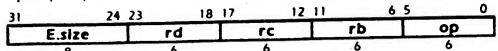
Selection

ciass	ор	type	size
	E.MUL	NONEM U P	8 16 32 64
multiply	E.MOL	C	8 16 32
multiply sum	E.MUL.SUM	NONEM U	8 16 32 o4
muluply sum	E.W.OE.SO	C	8 15 32
divide	E.DIV	NONEU	64

Format

E.op.size rd=rc,rb

rd=eopsize(rc,rb)



Description

Two values are taken from the contents of registers re and rb. The specified operation is performed, and the result is placed in register rd.

Definition

```
def mulisize,h,vs.v.i,ws.w.jl as muli \leftarrow ([vs&vsize-1+i]h-size | 1 | vsize-1+i.il * ([ws&wsize-1+j]h-size | 1 | wsize-1+j.il enddef | def c \leftarrow PolyMultiply(size,a,b) as p[0] \leftarrow 02*size for k \leftarrow 0 to size-1 p[k+1] \leftarrow p[k] * a_k ? (0size-k | 1 | b | 1 | 0k) : 02*size endfor c \leftarrow p[size]
```

```
enddef
```

```
def Ensemblelop.size.rd.rc.rb)
     c - RegReadirc, 128)
     b - RegReadith, 128)
     case on of
           EMUL: EMULC: EMULSUM. EMULSUM.C. E.CON. E.CON.C. E.DIV:
                 CS \leftarrow DS \leftarrow I
           EMULM; EMULSUMM, E.CON.M:
                 CI \leftarrow 0
                 bs ← 1
           E.MULU:, EMULSUM.U, E.CON.U, E.DIV.U, E.MUL.P.
                 cs \leftarrow bs \leftarrow 0
      endcase
      case op of
           E.MUL, E.MULU, E.MULM:
                 for i ← 0 to 64-size by size
                       d_{2^{n+size}-1...2^{n}} \leftarrow mul(size, 2^{n}size, cs.c.i, bs.b.i)
                 endfor
           E.MULP:
                 for i - 0 to 64-size by size
                       d2%size-1..2% ← PolyMultiply(size,csize-1+i,i-bsize-1+i,il-
                 endfor
            E.MULC:
                 for i - 0 to 64-size by size
                       if (i and size) = 0 then
                             p 		mul(size,2*size,1,=,i,1,b,i) - mul(size,2*size,1,c,i+size,1,b,i+size)
                             p ← mul(size,2*size,1,c,i,1,b,i+size) + mul(size,2*size,1,c,i,1,b,i+size)
                       endif
                       d24i+szej-1..24 ← P
                  endfor
            E.MULSUM, E.MULSUM.U, E.MULSUM.M:
                 P(0) - 0128
                 for i ← G to 128-size by size
                       p[+size] \leftarrow p[i] + mul(size, 128, cs, c, i, bs, b, i)
                  a ← p[128]
            EMULSUM.C:
                  p(0) ← 064
                  p|sze| ← 064
                  for i ← 0 to 128-size by size
                       if (i and size) = 0 then
                             p[\mapsto 2^*size] \leftarrow p[i] + mul[size,64,1,c,i,1,b,i]
                                                  - mul(size,64,1,c,i+size,1,b,i+size)
                        else
                             p[+2*size] \leftarrow p[i] + mul(size,64,1,c,i,1,b,i+size)
                                                 + mullsize,64,1,c,+size,1,b,i)
                        endf
                  endfor
                  a \leftarrow p[128 \cdot size] \mid 1 \mid p[128]
             E.CON, E.CON.U, E.CON.M.
                  P(0) ← 0128
                  for j ← 0 to 64-size by size
```

none

```
for i ← 0 to 64-size by size
                           P[j+size]24+size]-1...24 ← P[j]24+s-te]-1...24 *
                                 mulfsize, 2*size, cs.c, i+64-, bs.b./
                      endfor
                endfor
                a - p[64]
          E.CON.C:
                P(0) ← 0128
                for j ← 0 to 64-size by size
                      for i ← 0 to 64-size by size
                           if (|-i|) and j and size j=0 then
                                 p(j+size|2*j+size|-1.2*j ← PÜ|2*j+size|-1.2*j *
                                       mul(size, 2*size, 1, c, +64-j, 1, b, j)
                           eise
                                 p[j+size]24j+sizej-1..24 ← P[j]24j+sizej-1..24 *
                                       mul(size, 2*size, 1, c, i+64-j+2*size, 1, b, j)
                           endí
                      endfor
                endfor
                a - p|64]
          E.DIV:
                f(b) = 0) or ((c = [111063]) and (b = 164)) then
                      a - undefined
                else
                      q \leftarrow c / b
                      . ← c - q.p
                      a ← 163.0 11 963.0
                endif
          E.DIV.U:
                d b = 0 then
                      a ← undefined
                else
                      q ← 10 11 c1/10 11 b)
                      r ← c - 10 11 01*10 11 b)
                      a ← r<sub>63.0</sub> 11 q<sub>63.0</sub>
                endif
     endcase
     Reg'Arntefrd, 128, al
enddef
Exceptions
```

Ensemble Convolve Extract Immediate

These instructions take an address from a general register to fetch a large operand from memory, a second operand from a general register, perform a group of operations on partitions of bits in the operands, and catenate the results together, placing the result in a general register.

Operation codes

E.CON.KI.C.8.C.B	Ememble convolve cutract immediate signed complex bytes big-endian colling
E.CONXI.C.8.F.B	Ensemble convolve caract immediate signed complex bytes big-endam floor
E.CON.X.I.C.B.N.B	Ensemble convolve extract immediate signed complex bytes big endan nearest
E.CONX.I.C.8.Z.B	Ememble convolve extract immediate signed complex bytes big endum zero
E.CONXI.C.16.C.B	Ensemble convolve extract immediate signed complex doublets big-endian ceiling
E.CONXI.C. 16.F.B	Ememble convolve extract immediate signed complex doublets big endan floor
E.CONXI.C.16.N.B	Ememble convolve extract immediate signed complex doublets big-endian nearest
E.CCNXI.C.16.Z.B	Erriemble convolve extract immediate signed complex doublets big-cridian zero
E.CONX.I.C.32.C.B	Ensemble convolve extract immediate signed complex quadrets big-enchan resing
E.CONX.I.C.32.F.B	Ensymble convolve extract immediate signed complex quadlets big-endian floor
E.CONXI.C.32.N.B	Ensemble convolve extract immediate signed complex quadles big endian nearest
E.CONXI.C.32.Z.B	Ensemble convolve extract immediate signed complex quadlets big-endian zero
E.CONX.I.C.64.C.B	Ensemble convolve extract immediate signed complex ortics big endian ceiling
E CONXI.C.64.F.B	Ememble convolve ridract immediate signed complex octiess big endian floor
E.CONX.I.C.64.N.B	Ensemble convolve extract immediate signed complex actiess big-endian nearest
E.CONX.I.C.64.Z.B	Ensemble convolve extract animediate signed complex octions big endian zero
E.CONXI.C.8.C.L	Entemble convolve extract immediate signed complex bytes little-endian ceiling
E.CONXI.C.8.F.L	Ensemble convolve estruct immediate signed complex bytes little endian finor
E.CONXI.C.8.N.L	Ensemble convolve extract immediate signed complex bytes little endian nearest
E.CONX.I.C.8.Z.L	Ememble convolve extract immediate signed complex bytes little endian zero
E.CONX.I.C.16.C.L	Ensemble convolve extract immediate signed complim doublets little-endian colling
E.CON.X.I.C.16.F.L	Ensemble convolve extract immediate signed complex doublets little endian floor
E.CONX.I.C.16.N.L	Ensemble convolve extract immediate signed complex doubles; little endian neview
E.CONX.I.C.16.Z.L	Ensemble convolve extract immediate signed complex doublins little endusin zero
E.CONXI.C.32.C.L	Ememble convolve extract immediate signed complex quadlets little endian ceiling
E.CONX.I.C.32.F.L	Ensemble convolve extract immediate signed complex quadlets little endan floor
E.CONX.I.C.32.N.L	Ensemble convolve extract immediate signed complex quivalets little endian nearest
E.CONXI.C.32.Z.L	Ensemble convolve extract immediate signed complex quadlets little endian zero
E.CONX.I.C.64.C.L	Ensemble convolve extract immediate signed complex orders little endian ceiling
E.CONXI.C.64.F.L	Ensemble convolve extract immediate signed complex notices little endian floor
E.CONXI.C.64.N.L	Ensemble convolve entract immediate signed complex notices little endian nearest
E.CONX.i C.64.Z.L	Ensemble convolve extract immediate signed complex octiets little endian zero/
E.CONX.I.B.C B	Ensemble convolve extract immediate signed bytes big endian ceiling
E.CONX.I.8.F.B	Ensemble convolve extract immediate signed byces big endian floor
E.CON.X.I.S.N.B	Ensemble convolve extract immediate signed bytes big endian nearest
E.CON.X.I.8.Z.B	Ensemble convolve extract ammediate signed bytes big endian zero
E.CON.X.I.16.C.B	Ememble convolve extract ammediate signed doublets big endian ceiling

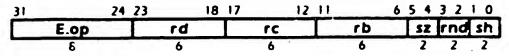
is course to	
E.CONXI.16.F.B	Ememble convolve citract immediate signed doublets big-endian floor
E.CON.X.I.16.N.B	Ememble convoive extract immediate signed doublets big-endian nearest
E.CON.X.I.16.Z.B	Ememble convolve citract immediate signed doubten big-endian zero
E.CON.X.I.32.C.B	Ememble convolve current immediate signed quadrets big-endian ceiling
E.CON.X.I.32.F.B	Ensemble convolve extract immediate signed quadicts big-endain floor
E.CON.X.I.32.N.B	Ememble convolve entract immediate signed quadlets big-endian nearest
E.CONXI.32.Z.B	Ememble convolve extract immediate signed quadlets big-endian zero
E.CON.X.I.64.C.B	Entermole convolve extract immediate signed or Cuts big endian ceiling
E.CON.X.I.64.F.B	Ensemble convolve extract immediate signed notices big-endian floor
E.CON.X.I.64.N.B	Enternale convolve extract immediate signed ucbus big-cris in nearest
E.CON.X.I.64.Z.B	Ensemble convolve extract immediate signed octice, big-crishin mediate
E.CON.X.I.8.C.L	Ensemble convolve extract immediate signed bytes Latte endian coding
E.CON.X.1.8.F.L	Ememble convolve extract immediate signar hyms into enuian floor
E.CON.X.I.8.N.L	Ensemble convolve cutract immediate lighted bytes little ensure incarest
E.CONXI.8.Z.L	Ensemble convolve extract immediate signed bytes little-endian zero
E.CONXI.16.C.L	Emerable convolve nitract immediate symed doublets little-endian ceiling
E.CONX.I.16.F.L	Ensemble convolve extract immediate signed doublets little-endum floor
E.CONX.I.16.N.L	Ensemble convolve extract insmediate signed doublets little-endian nearest
E.CONXI.16.Z.L	Ensemble convolve extract immediate signed doublets little-envisor zero
E.CON Y.I.32.C.L	Ensemble convolve extract immediate signed quadlets little enuian colling
E.CON.X32.F.L	Ensemble convolve extract immediate signed quidlets little Indian floor
E.CON.X.I.32.N.L	Ensemble convolve extract immediate signed quadlets little englan nearest
E.CON.X.I.32.Z.L	Ensemble convolve extract immediate signed quadlets little-endian zero
E.CON.X.I.64.C.L	Ensemble convolve extract immediate signed octions little enchan coding
E.CONXI.64.F.L	Ensemble convolve extract immediate signed outlets little-indian floor
E.CONXI.64.N.L	Ensemble convolve extract immediate signed octions little-endian nearest
E.CON.X.I.64.Z.L	Ensemble convolve extract immediate signed octiets little-endian zero
E.CON.X.I.M.8.C.B	Ensemble convolve extract immediate mixed signed bytes big-enduri ceiling
E.CONXI.M.8.F.B	Ensemble convolve extract mixed signed bytes big-endian floor
E.CONXI.M.8.N.B	Ensemble convolve extract immediate mixed signed bytes big endian nearest
E.CONX.I.M.8.Z.B	Ememble convolve extract in mediate mixed signed bytes big enrigin zero
E.CON.X.I.M.16.C.B	Ememble convolve extract immediate mixed-signed doublets big-endian ceiling
E.CONXI.M. 16.F.B	Ensemble convolve extract immediate mixed-signed doublets big-engian floor
E.CON.X.I.M. 16.P.B	Ememble convolve extract immediate mused-signed doublets big-endian nearest
E CONXI.M. 16.N.B	Ememble convolve citract immediate mixed-ugned doublets big-endian zero
E.C.C.N.X.I.M.32.C.B	Ensemble converve entract immediate mixed signed quadrets big-endian ceiling
E.CON.X.I.M.32.C.B	Ememble convolve extract immediate mixed-ugned quadlets big-enden floor
	Entemble convolve extract immediate mixed signed quadrets big-endian nearest
E.CONXIM.32.N.B	Ensemble convolve extract immediate mixed signed quadiets big-endian zero
E.CONXIM.32.Z.B	Ememble convolve intract immediate mixed signed quidates big-englan zero
E.CONXI.M.64.C.B	Ensemble convolve extract immediate mixed signed octoes big-ensists floor
E.CON.X.I.M.64.F.B	Ensemble convolve extract immediate mixed-ugned orders big-endian newest
E.CONXIM.64.N.B	
2.001.7.1.17.01.2.0	Ensemble convolve extract immediate mixed-signed octiets big-endian zero
E.CON.X.I.M.8.C.L	Enganths (concean and by amount to make the same to be a first to be a f
	Ensemble convolve extract immediate mixed signed bytes little endian ceiling
E.CON.X.I.M.8.F.L E.CON.X.I.M.8.N.L	Ensemble convolve extract immediate mixed signed bytes little endian ceiling Ensemble convolve extract immediate nixed-signed byte little-endian floor Ensemble convolve extract immediate mixed signed bytes little-endian nearest

E.CONXI.M.16.C.L ENOMAL.M.16.C.L ENOMA		
E.CONXI.M.16.F.L. Inventile convolve estirat immediate maind upped doublets little endian newest E.CONXI.M.16.N.L. Inventile convolve estirat immediate maind upped doublets little endian newest E.CONXI.M.32.C.L. Ensemble convolve estirat immediate maind upped doublets little endian newest e.CONXI.M.32.F.L. Ensemble convolve estirat immediate maind upped quadrets little endian ceiling E.CONXI.M.32.F.L. Ensemble convolve estirat immediate maind upped quadrets little endian ceiling E.CONXI.M.32.F.L. Ensemble convolve estirat immediate maind upped quadrets little endian newest E.CONXI.M.32.F.L. Ensemble convolve estirat immediate maind upped quadrets little endian newest E.CONXI.M.32.F.L. Ensemble convolve estirat immediate maind upped quadrets little endian newest E.CONXI.M.64.F.L. Ensemble convolve estirat immediate maind upped ocities little endian newest E.CONXI.M.64.F.L. Ensemble convolve estirat immediate maind upped ocities little endian newest e.CONXI.M.64.F.L. Ensemble convolve estirat immediate maind upped ocities little endian newest E.CONXI.M.64.F.L. Ensemble convolve estirat immediate unsigned bytes big endian ceiling E.CONXI.U.8.F.B. Ensemble convolve estirat immediate unsigned bytes big endian review E.CONXI.U.8.F.B. Ensemble convolve estirat immediate unsigned doublets big endian review E.CONXI.U.16.C.B. Ensemble convolve estirat immediate unsigned doublets big endian review E.CONXI.U.16.F.B. Ensemble convolve estirat immediate unsigned doublets big endian review E.CONXI.U.32.C.B. Ensemble convolve estirat immediate unsigned doublets big endian newest E.CONXI.U.32.F.B. Ensemble convolve estirat immediate unsigned doublets big endian newest E.CONXI.U.32.F.B. Ensemble convolve estirat immediate unsigned doublets big endian newest E.CONXI.U.64.F.B. Ensemble convolve estirat immediate unsigned doublets big endian newest E.CONXI.U.64.F.B. Ensemble convolve estirat immediate unsigned doublets big endian newest E.CONXI.U.64.F.B. Ensemble convolve estirat immediate unsigned doublets bittle endian flow E.CON	E.CONXI.M.8.Z.L	
E.CON.X.I.M.16.N.L E.CON.X.I.M.16.N.L E.CON.X.I.M.16.Z.L E.CON.X.I.M.32.C.L Enversible convenive estituet immediate mixed signed disablets little endain newer mixed signed disablets little endain new process of the convenive estituet immediate mixed signed quadrets little endain new process of the convenive estituet immediate mixed signed quadrets little endain new process of the convenive estituet immediate mixed signed quadrets little endain new process of the convenive estituet immediate mixed signed quadrets little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate mixed signed octies little endain new process of the convenive estituet immediate unsigned bytes big endain new process of the convenive estituet immediate unsigned bytes big endain new process of the convenive estituet immediate unsigned disablets big endain new process of the convenive estituet immediate unsigned disablets big endain new process of the convenive estituet immediate unsigned disablets big endain new process of the convenive estituet immediate unsigned disablets big endain newest process of the convenive estituet immediate unsigned disablets big endain newest process of the convenive estituet immediate unsigned disablets big endain newest process of the convenive estituet immediate unsigned disablets big endain newest process of the convenive estituet immediate unsigned disablets big endain newest process of the convenive estituet immediate unsigned disablets big endain newest process of the convenive estituet immediate unsigned disablets l	E.CON.X.I.M.16.C.L	
E.CONX.I.M.16.2.L Intermite convolve estivat ammediate mixed signed doublets lettle endian zero E.CONX.I.M.32.F.L Intermite convolve estivat ammediate mixed signed quadrets lettle endian ceiling E.CONX.I.M.32.F.L Enversible convolve estivat ammediate mixed signed quadrets lettle endian nov. est E.CONX.I.M.32.R.L Enversible convolve estivat ammediate mixed signed quadrets lettle endian nov. est E.CONX.I.M.32.R.L Enversible convolve estivat ammediate mixed signed quadrets lettle endian nov. est E.CONX.I.M.64.F.L Enversible convolve estivat ammediate mixed signed outlets lettle endian zero est E.CONX.I.M.64.R.L Enversible convolve estivat ammediate mixed signed outlets lettle endian zero except estivat ammediate mixed signed outlets lettle endian zero except except estivat ammediate mixed signed outlets lettle endian zero except except estivat ammediate mixed signed outlets lettle endian zero except except extinat ammediate mixed signed outlets lettle endian zero except except except extinat ammediate mixed signed outlets lettle endian zero except except except extinat ammediate unsigned bytes big endian review except except extinat ammediate unsigned bytes big endian zero except except except extinat ammediate unsigned bytes big endian zero except except except extinat ammediate unsigned doublets big endian zero except except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned doublets big endian zero except except extinat ammediate unsigned double	E.CONX.I.M. 16.F.L	
E.CON.X.I.M.32.C.L Ensemble convolve estituet immediate materi signed quadrets little endum coloring E.CON.X.I.M.32.C.L Ensemble convolve estituet immediate materi signed quadrets little endum nowest excessional and signed quadrets little endum nowest excessional and signed quadrets little endum nowest excessional and signed extress little endum nowest excessional and signed extress little endum nowest excessional excessional excessional entress little endum nowest excessional excessional excessional entress little endum excessional ex	E.CON.X.I.M. 16.N.L	Ensemble convolve extract immediate mixed signed doublets little endian nearest
E.CONX.I.M.32.F.L Ensemble convolve extract immediate maked signed quadrets little endain now extract. Immediate maked signed outlets little endain now extract. Immediate unsigned signed outlets little endain now extract. Immediate unsigned bytes big endain now extract. Immediate unsigned bytes big endain now extract. Immediate unsigned doublets big endain now extract. Immediate unsigned quadrets big endain now extract. Immediate unsigned doublets big endain now extract. Immediate unsigned doublets big endain now extract. Immediate unsigned publics little endain now extract. Immediate unsigned publics little endain now extract. Immediate unsigned doublets little endain now extrac	E.CONXI.M.16 Z.L	Ensemble convolve extract immediate mixed signed distillers little endual zero
E.CONX.I.M.32.N.L E.CONX.I.M.32.N.L E.CONX.I.M.32.N.L E.CONX.I.M.32.N.L E.CONX.I.M.32.N.L E.CONX.I.M.32.N.L E.CONX.I.M.32.N.L E.CONX.I.M.64.C.L Ensemble is estiract immediate mated signed quadlets little endian ceiling E.CONX.I.M.64.R.L E.CONX.I.M.64.N.L E.CONX.I.M.66.N.L E.CONX.I.M.66	E.CON.X.I.M.32.C.L	Ensemble convolve extract immediate mixed signed quadrets little endian colleg-
E.CONX.I.M.32.Z.L Ensemble is extract immediate mixed signed quadlets little endan zero in extract immediate mixed signed octies little endan ceiling in extract immediate mixed signed octies little endan ceiling in extract immediate mixed signed octies little endan review in extract immediate mixed signed octies little endan review in extract immediate mixed signed octies little endan review in extract immediate mixed signed octies little endan review in extract immediate mixed signed octies little endan review in extract immediate mixed signed octies little endan review in extract immediate imagened bytes lightly in endan ceiling in extract immediate unsigned bytes lightly endan review in extract immediate unsigned bytes lightly endan review in extract immediate unsigned doublets lightly endan review in endance convolve extract immediate unsigned doublets lightly endance review in endance endance lightly endance endance lightly endance endance lightly endance endance lightly endance lightly endance endance li	E.CON.X.I.M.32.F.L	
E.CONX.I.M.64.C.L Ensemble is extract ammediate mated signed orders little endian colling E.CONX.I.M.64.P.L Ensemble convolve extract ammediate mated signed orders little endian newest E.CONX.I.M.64.P.L Ensemble convolve extract ammediate mated signed orders little endian newest E.CONX.I.M.64.P.L Ensemble convolve extract ammediate mated signed orders little endian newest E.CONX.I.U.8.C.B Ensemble convolve extract ammediate mated signed orders little endian newest E.CONX.I.U.8.P.B Ensemble convolve extract ammediate unsigned bytes big endian floor E.CONX.I.U.8.P.B Ensemble convolve extract ammediate unsigned bytes big endian newest E.CONX.I.U.16.C.B Ensemble convolve extract ammediate unsigned doubtes big endian review E.CONX.I.U.16.P.B Ensemble convolve extract ammediate unsigned doubtes big endian review E.CONX.I.U.16.P.B Ensemble convolve extract ammediate unsigned doubtes big endian newest E.CONX.I.U.32.C.B Ensemble convolve extract ammediate unsigned doubtes big endian newest E.CONX.I.U.32.P.B Ensemble convolve extract ammediate unsigned quadrets big endian newest E.CONX.I.U.32.P.B Ensemble convolve extract ammediate unsigned quadrets big endian newest E.CONX.I.U.64.C.B Ensemble convolve extract ammediate unsigned distinct big endian newest E.CONX.I.U.64.P.B Ensemble convolve extract ammediate unsigned extress big-endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned extress big-endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned displets little endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned displets	E.CON.X.I.M.32.N.L	Ensemble convolve extract immediate mixed signs, quadiets little endian newest
E.CONX.I.M.64.P.L Insemble convolve extract ammediate mixed signed orders little endian newest E.CONX.I.M.64.P.L Insemble convolve extract ammediate mixed signed orders little endian newest E.CONX.I.M.64.P.L Insemble convolve extract ammediate mixed signed orders little endian newest E.CONX.I.U.8.C.B Insemble convolve extract ammediate unsigned bytes big endian floor E.CONX.I.U.8.P.B Insemble convolve extract ammediate unsigned bytes big endian newest E.CONX.I.U.8.P.B Insemble convolve extract ammediate unsigned doublets big endian newest E.CONX.I.U.16.C.B Insemble convolve extract ammediate unsigned doublets big endian review E.CONX.I.U.16.P.B Insemble convolve extract ammediate unsigned doublets big endian newest E.CONX.I.U.16.P.B Insemble convolve extract ammediate unsigned doublets big endian review E.CONX.I.U.16.P.B Insemble convolve extract ammediate unsigned doublets big endian newest E.CONX.I.U.32.P.B Insemble convolve extract ammediate unsigned quadrets big endian newest E.CONX.I.U.32.P.B Insemble convolve extract ammediate unsigned quadrets big endian newest E.CONX.I.U.64.P.B Ensemble convolve extract ammediate unsigned diablets big endian newest E.CONX.I.U.64.P.B Ensemble convolve extract ammediate unsigned extress big endian floor E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned extress big endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned extress big endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.8.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned orders big endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned doublets little endian newest E.CONX.I.U.16.P.L Ensemble convolve extract ammediate unsigned doublets littl	E.CON.X.I.M.32.Z.L	
E.CONX.I.M.64.N.L Ensemble convolve extract immediate mixed signed orders little endian newest incompleted in the endian newest in the endian newest incompleted in the endian ending endian endian ending endian ending endian ending endian endian ending endian ending endian endian endian ending en	E.CON.X.I.M.64.C.L	
E.CON.X.I.M.64.Z.L. Ensemble convolve extract immediate unsigned orders lettle endian zivo. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned bytes big endian ceiling. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned bytes big endian floor. E.CON.X.I.U.B.N.B. Ensemble convolve extract immediate unsigned doublets big endian ceiling. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets big endian ceiling. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets big endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets big endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned quadlets big endian ceiling. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned quadlets big endian ceiling. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned quadlets big endian ceiling. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned quadlets big endian reviest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned octies big endian reviest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned octies big endian reviest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned octies big endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned plates little endian floor. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned bytes little endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets little endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets little endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets little endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned doublets little endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned disablets little endian nearest. E.CON.X.I.U.B.F.B. Ensemble convolve extract immediate unsigned disablet	E.CONX.I.M 64.F.L	
E.CON.X.I.U.8.C.B Ensemble convolve extract immediate unsigned bytes big endian ceilins) E.CON.X.I.U.8.F.B Ensemble convolve extract immediate unsigned bytes big endian floor E.CON.X.I.U.8.N.B Ensemble convolve extract immediate unsigned doublets big endian ceiling in the property of	E.CON.X.I.M.64.N.L	Ensemble re-waive extract immediate mixed signed actions little endian nearest
E:CONX.I.U.8.R.B Ensemble convolve extract immediate unsigned bytes big endian floor E:CONX.I.U.8.N.B Ensemble convolve extract immediate unsigned doublets big endian realiest E:CONX.I.U.16.C.B Ensemble convolve extract immediate unsigned doublets big endian realiest E:CONX.I.U.16.R.B Ensemble convolve extract immediate unsigned doublets big endian floor E:CONX.I.U.16.N.B Ensemble convolve extract immediate unsigned quadlets big endian reviest E:CONX.I.U.32.C.B Ensemble convolve extract immediate unsigned quadlets big endian reviest E:CONX.I.U.32.R.B Ensemble convolve extract immediate unsigned quadlets big endian reviest E:CONX.I.U.32.N.B Ensemble convolve extract immediate unsigned quadlets big endian reviest E:CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned octions big endian reviest E:CONX.I.U.64.R.B Ensemble convolve extract immediate unsigned octions big endian newest E:CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian newest E:CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian newest E:CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian newest E:CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian newest E:CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.8.R.L Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.6.C.L Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.6.R.L Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.6.R.L Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.6.R.L Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.6.R.L Ensemble convolve extract immediate unsigned doublets little endian newest E:CONX.I.U.6.R.L Ensemble convolve extract immediate unsign	E.CON.X.I.M.64.Z.L	
E.CONX.I.U.8.N.B Ensemble convolve extract immediate unsigned bytes big endian nearest E.CONX.I.U.16.C.B Ensemble convolve extract immediate unsigned doublets big endian college E.CONX.I.U.16.F.B Ensemble convolve extract immediate unsigned doublets big endian flow E.CONX.I.U.16.N.B Ensemble convolve extract immediate unsigned doublets big endian nearest E.CONX.I.U.3.2.C.B Ensemble convolve extract immediate unsigned quadlets big endian college E.CONX.I.U.3.2.F.B Ensemble convolve extract immediate unsigned quadlets big endian floor E.CONX.I.U.3.2.N.B Ensemble convolve extract immediate unsigned quadlets big endian nearest E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned octions big endian floor E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned octions big endian nearest E.CONX.I.U.64.R.B Ensemble convolve extract immediate unsigned octions big endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octions big endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned distributes little endian nearest E.CONX.I.U.6.C.L Ensemble convolve extract immediate unsigned distributes little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned distributes little endian nearest E.CONX.I.U.3.2.C.L Ensemble convolve extract immediate unsigned distributes little endian nearest E.CONX.I.U.3.2.C.L Ensemble convolve extract immediate unsigned distributes little endian nearest E.CONX.I.U.3.2.N.L Ensemble convolve extract immediate unsigned distributes little endian nearest E.CONX.I.U.6.4.C.L Ensemble convolve extract immediate unsigned octions little endian nearest E.CONX.I.U.6.4.C.L Ensemble convolve extract immediate unsigned octions little endian nearest E.CONX.I.U.6.4.	E.CON.X.I.U.8.C.B	
E.CONX.I.U.16.C.B Ensemble convolve extract immediate unsigned doublets big endian colleg E.CONX.I.U.16.F.B Ensemble convolve extract immediate unsigned doublets big endian floir E.CONX.I.U.16.N.B Ensemble convolve extract immediate unsigned doublets big endian newest E.CONX.I.U.32.C.B Ensemble convolve extract immediate unsigned quadiets big endian ceiling E.CONX.I.U.32.F.B Ensemble convolve extract immediate unsigned quadiets big endian ceiling E.CONX.I.U.32.N.B Ensemble convolve extract immediate unsigned duadiets big endian newest E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned oxites big endian newest E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned oxites big endian newest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned oxites big-endian review E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned oxites big-endian newest E.CONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes liftle endian ceiling E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned doublets liftle endian nearest E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned doublets liftle endian ceiling E.CONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets liftle endian ceiling E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets liftle endian reviest E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets liftle endian reviest E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets liftle endian reviest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets liftle endian reviest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned oxites liftle endian reviest E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned oxites liftle endian reviest E.CONX.I.U.64.F.L	E.CON.X.I.U.8.F.B	
E.CONX.I.U.16.R.B Ensemble convolve extract immediate unsigned doublets big endian flow. E.CONX.I.U.16.N.B Ensemble convolve extract immediate unsigned quadlets big endian review. E.CONX.I.U.32.C.B Ensemble convolve extract immediate unsigned quadlets big endian ceiling. E.CONX.I.U.32.R.B Ensemble convolve extract immediate unsigned quadlets big endian floor. E.CONX.I.U.32.N.B Ensemble convolve extract immediate unsigned quadlets big endian newest. E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned octies big-endian review. E.CONX.I.U.64.R.B Ensemble convolve extract immediate unsigned octies big-endian floor. E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octies big-endian newest. E.CONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian floor. E.CONX.I.U.8.R.L Ensemble convolve extract immediate unsigned bytes little endian floor. E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned bytes little endian coling. E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian coling. E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian floor. E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian newest. E.CONX.I.U.3.2.C.L Ensemble convolve extract immediate unsigned quadlets little endian coling. E.CONX.I.U.3.2.N.L Ensemble convolve extract immediate unsigned quadlets little endian newest. E.CONX.I.U.3.2.N.L Ensemble convolve extract immediate unsigned quadlets little endian newest. E.CONX.I.U.3.2.N.L Ensemble convolve extract immediate unsigned quadlets little endian newest. E.CONX.I.U.6.4.C.L Ensemble convolve extract immediate unsigned octies little endian newest. E.CONX.I.U.6.4.C.L Ensemble convolve extract immediate unsigned octies little endian floor. E.CONX.I.U.6.4.C.L Ensemble convolve extract immediate unsigned octies little endian floor.	E.CON.X.I.U.8.N.B	
ECONX.I.U.32.C.B Ensemble convolve extract immediate unsigned disablets big endain necest ECONX.I.U.32.F.B Ensemble convolve extract immediate unsigned quadlets big endain review ECONX.I.U.32.F.B Ensemble convolve extract immediate unsigned quadlets big endain nearest ECONX.I.U.32.N.B Ensemble convolve extract immediate unsigned quadlets big endain nearest ECONX.I.U.64.C.B Ensemble convolve extract immediate unsigned octies big-endain review ECONX.I.U.64.F.B Ensemble convolve extract immediate unsigned octies big-endain nearest ECONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octies big-endain nearest ECONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endain floor ECONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endain nearest ECONX.I.U.8.N.L Ensemble convolve extract immediate unsigned doublets little endain nearest ECONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endain nearest ECONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endain nearest ECONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endain nearest ECONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endain nearest ECONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endain nearest ECONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endain nearest ECONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endain nearest ECONX.I.U.32.R.L Ensemble convolve extract immediate unsigned quadlets little endain nearest ECONX.I.U.34.C.L Ensemble convolve extract immediate unsigned octiets little endain nearest ECONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octiets little endain nearest ECONX.I.U.64.F.L Ensemble convolve extract immediate unsigned octiets little endain nearest ECONX.I.U.64.F.L	E.CON.X.I.U. 16.C.B	
E.CONX.I.U.32.C.B Ensemble convolve extract immediate unsigned quadlets big endian colong ECONX.I.U.32.F.B Ensemble convolve extract immediate unsigned quadlets big endian floor E.CONX.I.U.32.N.B Ensemble convolve extract immediate unsigned quadlets big endian nearest E.CONX.I.U.64.C.B Ememble convolve extract immediate unsigned extless big-endian ceiling E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned extless big-endian floor E.CONX.I.U.64.P.B Ensemble convolve extract immediate unsigned extless big-endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned bytes little endian ceiling E.CONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian floor E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian floor E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian review E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian review E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian review E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian review E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned octies little endian review E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned octies little endian floor E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned octies little endian floor	E.CON.X.I.U.16.F.B	
ECONX.I.U.32.F.B Ensemble convolve extract immediate unsigned quadlets big endian floor ECONX.I.U.32.N.B Ensemble convolve extract immediate unsigned quadlets big endian newest E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned octies big endian review E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned octies big endian floor E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octies big-endian newest E.CONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian colling E.CONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian colling E.CONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian floor E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian floor E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian review E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian review E.CONX.I.U.64.F.L	E CONX.I.U. 16.N.B	
E.CONX.I.U.64.C.B Ensemble convolve extract immediate unsigned quadlets big endian nearest E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned extress big-endian coling E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned extress big-endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned extress big-endian nearest E.CONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian coling E.CONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian invirest E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian invirest E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian coling E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian coling E.CONX.I.U.32.R.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian coling E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octions little endian coling E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned octions little endian coling E.CONX.I.U.64.F.L	E.CON.X.I.U.32.C.B	
E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned octies big-endian colling E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned octies big-endian nearest E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octies big-endian nearest E.CONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian ceiling E.CONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian colling E.CONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian inexest E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian colling E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian colling E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned octies little endian colling E.CONX.I.U.64.F.L	E CONXI.U.32.F.B	
E.CONX.I.U.64.F.B Ensemble convolve extract immediate unsigned octies big-endian floor E.CONX.I.U.64.N.B Ensemble convolve extract immediate unsigned octies big-endian nearest E.CONX.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian ceiling E.CONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian ceiling E.CONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian ceiling E.CONX.I.U.32.R.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned octies little endian nearest E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian ceiling	E.CON.X.I.U.32.N.B	Ensemble convolve citract immediate unsigned quadlets big endian nearest
E.CON.X.I.U.8.C.L Ensemble convolve extract immediate unsigned orties big-endian nearest E.CON.X.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian colling E.CON.X.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CON.X.I.U.8.N.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CON.X.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian colling E.CON.X.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian floor E.CON.X.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest E.CON.X.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian colling E.CON.X.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned extract little endian ceiling E.CON.X.I.U.64.C.L Ensemble convolve extract immediate unsigned extress little endian ceiling E.CON.X.I.U.64.F.L Ensemble convolve extract immediate unsigned extress little endian floor	E.CON.X.I.U.64.C.B	
E.CON.X.I.U.8.C.L Ensemble convolve extract immediate unsigned bytes little endian colling E.CON.X.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CON.X.I.U.8.N.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CON.X.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian colling E.CON.X.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian linearest E.CON.X.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian linearest E.CON.X.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian colling E.CON.X.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CON.X.I.U.34.C.L Ensemble convolve extract immediate unsigned octiets little endian colling E.CON.X.I.U.64.C.L Ensemble convolve extract immediate unsigned octiets little endian floor E.CON.X.I.U.64.F.L Ensemble convolve extract immediate unsigned octiets little endian floor	E.CON.X.I.U.64.F.B	
ECONX.I.U.8.F.L Ensemble convolve extract immediate unsigned bytes little endian floor E.CONX.I.U.8.N.L Ensemble convolve extract immediate unsigned bytes little endian nearest E.CONX.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian colling E.CONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian floor E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian reviest E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian colling E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.34.C.L Ensemble convolve extract immediate unsigned octies little endian ceiling E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian ceiling	E.CON.X.I.U.64.N.B	
E.CON.X.I.U.8.N.L Ensemble convolve extract immediate unsigned bytes little endian nearest ECON.X.I.U.16.C.L Ensemble convolve extract immediate unsigned doublets little endian colling ECON.X.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian nearest ECON.X.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian nearest ECON.X.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian colling ECON.X.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor ECON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest ECON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest ECON.X.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian colling E.CON.X.I.U.64.F.L Ensemble convolve extract immediate unsigned octies little endian floor	E.CONX.I.U.8.C.L	
E.CON.X.I.U.16.C.L. Ensemble convolve extract immediate unsigned doublets little endian colling. E.CON.X.I.U.16.F.L. Ensemble convolve extract immediate unsigned doublets little endian floor. E.CON.X.I.U.16.N.L. Ensemble convolve extract immediate unsigned doublets little endian reviest. E.CON.X.I.U.32.C.L. Ensemble convolve extract immediate unsigned quadlets little endian colling. E.CON.X.I.U.32.F.L. Ensemble convolve extract immediate unsigned quadlets little endian floor. E.CON.X.I.U.32.N.L. Ensemble convolve extract immediate unsigned quadlets little endian newest. E.CON.X.I.U.32.N.L. Ensemble convolve extract immediate unsigned octies little endian ceiling. E.CON.X.I.U.64.C.L. Ensemble convolve extract immediate unsigned octies little endian floor.	E CONX.I.U.8.F.L	Ensemble convolve extract immediate unsigned bytes little endian floor
ECONX.I.U.16.F.L Ensemble convolve extract immediate unsigned doublets little endian finor E.CONX.I.U.16.N.L Ensemble convolve extract immediate unsigned doublets little endian reviest E.CONX.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian coling E.CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian finor E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned octies little endian ceiling E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian ceiling E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned octies little endian floor	E.CON.X.I.U.8.N.L	
E.CON.X.I.U.16.N.L Ensemble convolve catract immediate unsigned doublets little endian nearest E.CON.X.I.U.32.C.L Ensemble convolve extract immediate unsigned quadlets little endian colling E.CON.X.I.U.32.F.L Ensemble convolve extract immediate unsigned quadlets little endian floor E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned quadlets little endian nearest E.CON.X.I.U.32.N.L Ensemble convolve extract immediate unsigned octies little endian nearest E.CON.X.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian floor	E CON X.I.U. 16.C.L	l
E.CON.X.I.U.32.C.L Ensemble converve extract immediate unsigned quadlets little endian colling ECON.X.I.U.32.F.L Ensemble converve extract immediate unsigned quadlets little endian floor E.CON.X.I.U.32.N.L Ensemble converve extract immediate unsigned quadlets little endian nearest E.CON.X.I.U.32.N.L Ensemble converve extract immediate unsigned octies little endian ceiling E.CON.X.I.U.64.C.L Ensemble convolve extract immediate unsigned octies little endian floor	E CONX.I.U. 16.F.L	
E CONX.I.U.32.F.L Ensemble convolve extract immediate unsigned quadrets little endian floor E.CONX.I.U.32.N.L Ensemble convolve extract immediate unsigned quadrets little endian nearest E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned extress little endian ceiling E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned extress little endian floor	E.CONX.I.U.16.N.L	
E.CON.X.I.U.32.N.L Ensemble convolve extract ammediate unsigned quadlets little endian nearest E.CON.X.I.U.64.C.L Ensemble convolve extract ammediate unsigned extres little endian ceiling E.CON.X.I.U.64.F.L Ensemble convolve extract ammediate unsigned extres little endian floor	E CON X.I.U.32.C.L	
E.CONX.I.U.64.C.L Ensemble convolve extract immediate unsigned acties little endian ceiling E.CONX.I.U.64.F.L Ensemble convolve extract immediate unsigned acties little endian floor	E CON.X.I.U.32.F.L	
E.CON.X.I.U.64.F.L Ensemble convolve extract immediate unsigned orders little endian floor	E.CONX.I.U.32.N.L	
E.CONA.I.O.64.F.C	E CONX.I.U.64.C.L	
E CONX.I.U.64 N.L. Ensemble convolve extract immediate unsigned orticls little endian nearest	E.CONX.I,U.64.F.L	
	E CON.X.I.U.64 N.L	Ensemble convolve extract immediate unsigned orticls little endian nearest

Format

E.op.size.rnd rd@rc,rb,i

rd=eopsizernd(rd,rc,rb,i)



$$sz \leftarrow log(size) - 3$$

 $sh \leftarrow size + 7 - log(size) - i$

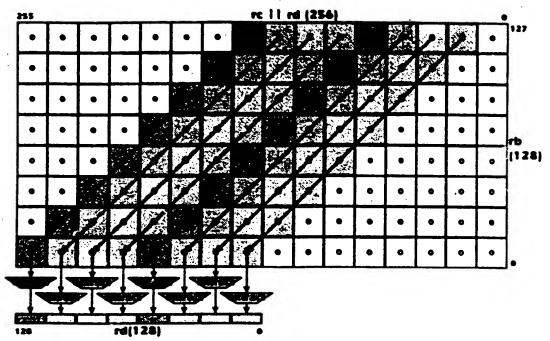
Description

The contents of registers rd and rc are catenated, as specified by the order parameter, and used as a first value. A second value is the contents of register rb. The values are partitioned into groups of operands of the size specified and are convolved, producing a group of values. The group of values is rounded, and limited as specified, yielding a group of results which is the size specified. The group of results is catenated and placed in register rd.

Z (zero) rounding is not defined for unsigned extract operations, and a ReservedInstruction exception is raised if attempted. F (floor) rounding will properly round unsigned results downward.

The order parameter of the instruction specifies the order in which the contents of registers rd and rc are catenated. The choice is significant because the contents of register rd is overwritten. When little-endian order is specified, the contents are catenated so that the contents of register rc is most significant (left) and the contents of register rd is least significant (right). When big-endian order is specified, the contents are catenated so that the contents of register rd is most significant (left) and the contents of register rc is least significant (right).

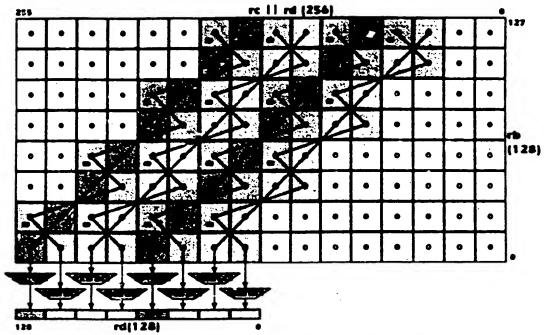
An ensemble-convolve-extract-immediate-doublets instruction (ECON.X.116, ECON.X.1M16, or ECON.X.1U16) convolves vector [x w v u t s r q p o n m l k j i] with vector [h g f e d c b a], yielding the products [ax+bw+cv+du+et+fs+gr+hq ... as+br+cq+dp+eo+fn+gm+hl ar+bq+cp+do+en+fm+gl+hk aq+bp+co+dn+em+fl+gk+hj], rounded and limited as specified:



Ensemble convolve extract immediate doublets

Ensemble Convolve Extract Immediate

An ensemble-convolve-extract-immediate-complex-doublets instruction (ECON.X.IC16) convolves vector [x w v u t s r q p o n m l k j i] with vector [h g f e d c b a], yielding the products [ax+bw+cv+du+et+fs+gr+hq ... as-bt+cq-dr+eo-fp+gm-hn ar+bq+cp+do+en-fm+gl+hk aq-br+co-dp+em-fn+gk+hl], rounded and limited as specified.



Ensemble convolve extract immediate complex doublets

Definition

```
def mul(size,h,vs,v,i,ws,w,j) as
     mul ← ((vs&vsze-1+1)h-sze 11 vsze-1+1,1 * ((ws&wsze-1+1)h-sze 11 wsze-1+1,1)
enddef
def EnsembleConvolveExtractImmediate(op.rnd,gsize,rd,rc,rb,sh)
     d ← RegRead(rd, 128)
     c ← RegRead(rd, 128)
     b ← RegRead(rb, 128)
     Igsize - logigsize
     wsize ← 128
     msize ← 256
     vsize ← 129
     case op of
          E.CON.X.I.B, E.CON.X.I.U.B, F.CON.X.I.M.B, E.CON.X.I.C B:
                m \leftarrow d \sqcap c
          E.CONXIL, E.CONXI.U.L E.CONXI.M.L E.CONXI.C.L:
                m \leftarrow c H d
     endcase
     case op of
          E.CON.X.I.U.B, E.CON.X.I.U.L:
                as \leftarrow ms \leftarrow bs \leftarrow false
          E.CON.X.I.M.B, E.CON.X.I.M.L:
```

```
ms - false
                   as ← bs ← true
             E.CONXI.B. E.CONXI.L. E.CONXI.C.B. E.CONXI.C.L:
                   as ← ms ← bs ← true
       endcase
       h \leftarrow (2^{\circ}gsze) + 7 - igsze
       r \leftarrow h \cdot size \cdot sh
       for i \leftarrow 0 to wsize-gsize by gsize
             q(0) \leftarrow 0^{2^{n}}gsize^{-7}-Igsize
             for j \leftarrow 0 to vsize-gsize by gsize
                   case op of
                         E.CONXI.B. E.CONXI.L E.CONXI.M.B. E.CONXI.M.L.
                         E.CONXI.U.B. E.CONXI.U.L.
                              q[+gsize] ← q[j] + mul(gsize,h,ms,m,i+128-j,bs,b,f)
                        E.CONXI.C.B. E.CONXI.C.L:
                              if (-1) & 1 & gsize = 0 then
                                    q[+gsize] ← q[j] + mul(gsize,h,ms,m,i+128-j,bs,b,j)
                                    alj+gsizej ← alj] - mullgsize,h,ms,m,i+128-j+2*gsize,bs,b,ij
                              endif
                  endcase
            endfor
            p ← glysizel
            case rnd of
                  none, N:
                       s \leftarrow 0^{h-r} \mid 1 \cdot p_r \mid 1 \mid p_r^{r-1}
                 Z:
                       s ← 0h-r 1 ph-1
                 F:
                       s ← 0h
                 C:
                       s ← 0h-r 11 1r
           endcase
           v ← ((as & ph.1)11p) + (011s)
           # V_{h,r+gsize} = las & v_{r+gsize-1}^{h+1-r-gsize} then
                 agsize-1+i.i ← Vgsize-1+r.r
           else
                 a_{gsize-1+...i} \leftarrow as 7 (v_h 11 - v_R^{size-1}) : 19 size
           endif
     endfor
     2127. wsize ← 0
     RegWrite(rd. 128, a)
enddel
```

Exceptions

none

Ensemble Convolve Floating-point

These instructions take an address from a general register to tetch a larve operand from memory, a second operand from a general register, perform a group of operations on partitions of bits in the operands, and catenate the results together, placing the result in a general register.

Operation codes

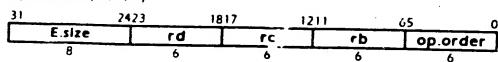
E.CON.F. 16.B	Ensemble convolve floating-point half big-endian
E.CON.F. 16.L	Ensemble convolve floating-point half little-endian
E.CON.7.32.B	Ensemble convolve floating-point single big-endian
E.CON.F.32.L	Ensemble convolve floating-point single little-endian
E.CON.F.64.B	Ensemble convolve floating-point double big-endian
E.CON.F.64.L	Ensemble convolve floating-point double little-endian
E.CON.C.F.16.B	Ensemble convolve complex floating-point half big-endian
E.CON.C.F.16.L	Ensemble convolve complex floating-point half little-encian
E.CON.C.F.32.B	Ensemble convolve complex floating-point single big-endian
E.CON.C.F.32.L	Ensemble convolve complex floating-point single little-endian
E.CON.C.F.64.B	Ensemble convolve complex floating-point double big-endian
E.CON.C.F.64.L	Ensemble convolve complex floating-point double little-endian

Format

E.op.size.order

rd=rc.rb

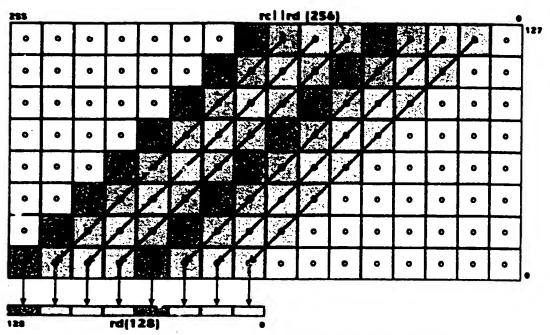
rd=eopsizeorder(rd,rc,rb)



Description

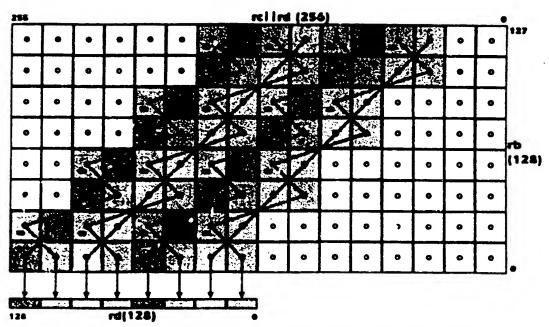
The first value is the catenation of the contents of register rd and rc, as specified by the order parameter. A second value is the contents of register rb. The values are partitioned into groups of operands of the size specified. The second values are multiplied with the first values, then summed, producing a group of result values. The group of result values is catenated and placed in register rd.

An ensemble-convolve-floating-point-half-little-endian instruction (E.CON.F.16.L) convolves vector [x w v u t s r q p o n m l k j i] with vector [h g f e d c b a], yielding the products [ax+bw+cv+du+et+fs+gr+hq ... as+br+cq+dp+eo+fn+gm+hl ar+bq+cp+do+en+fm+gl+hk aq+bp+co+dn+em+fl+gk+hj]:



Ensemble convolve floating-point half little-endian

A ensemble-convolve-complex-floating-point-half-little-endian instruction (E.CON.C.F.16.L) convolves vector [x w v u t s r q p o n m l k j i] with vector [h g f e d c b a], yielding the products [ax+bw+cv+du+ct+fs+gr+hq ... as-bt+cq-dr+eo-fp+gm-hn ar+bq+cp+do+en+fm+gl+hk aq-br+co-dp+em-fn+gk+hl]:



Ensemble convolve complex floating-point half little-endian

Definition

```
def mul(size,v,i,w,t) as
     mul - fmul(F(size, v<sub>size-1+1.</sub>),F(size, w<sub>size-1+1.</sub>))
enddef
del EnsembleConvolveFloatingPoint(op.qsize,rd,rc,rh)
     d - RegRead(rd, 128)
     c ← RegReadfrc, 128)
     b ← RegRead(rb, 128)
     Igsize - logigsize
     wsize ← 128
     msize ← 256
     vsize ← 128
     case op of
          E.CON.F.B. E.CON.C.F.B:
                m \leftarrow d l l c
          E.CON.F.L. E.CON.C.F.L:
                m ← c II d
     endcase
    for i \leftarrow 0 to wsize-gsize by gsize
          //NULL value doesn't combine with zero to alter sign bit
          q|0|.t ← NULL
          for j \leftarrow 0 to vsize-gsize by gsize
               case op of
                     E.CONF.L. E.CONF.B:
```

none

```
| alj+gsize| ← fadd(alj|, mul(gsize,m,i+128-j,b,j|) | E.CONCF.B: | if (-i) & j & gsize = 0 then | alj+gsize| ← fadd(alj|, mul(gsize,m,i+128-j,b,j|) | else | alj+gsize| ← fsub(alj|, mul(gsize,m,i+128-j+2*gsize,b,j|) | endif | endcase | endfor | agsize-1+...i ← PackF(gsize,alvsize|,N) | andfor | alizn.wsize ← 0 | RegWrite(rd, 128, alj enddef | Exceptions
```

Ensemble Extract

These operations take operands from three registers, perform operations on partitions of bits in the operands, and place the concatenated results in a fourth register.

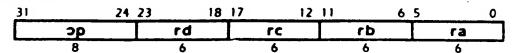
Operation codes

E.MUL.X	Ensemble multiply extract
E.EXTRACT	Exisemble extract
E.SCALADD.X	Ensemble scale add extract

Format

E.opra=rd,rc,rb

ra=gop(rd,rc,rb)



Description

The contents of registers rd, rc, and rb are fetched. The specified operation is performed on these operands. The result is placed into register ra.

Bits 31.0 of the contents of register rb specifies several parameters which control the manner in which data is extracted, and for certain operations, the manner in which the operation is performed. The position of the control fields allows for the source position to be added to a fixed control value for dynamic computation, and allows for the lower 16 bits of the control field to be set for some of the simpler extract cases by a single GCOPYI.128 instruction. The control fields are further arranged so that if only the low order 8 bits are non-zero, a 128-bit extraction with truncation and no rounding is performed.

31	24 23		16	1514131211109 8		0
fsize		dpos		x s nm l rnd	gssp	
8		8		111112	9	

The table below describes the meaning of each label:

label	bits	meaning
fsize	8	field size
dpos	8	destination position
x	1	reserved
S	l i	signed vs. unsigned
n	11	complex vs. real multiplication
m	1	merge vs. extract or mixed-sign vs. same-sign multiplication
1	11	limit: saturation vs. truncation
rnd	12	rounding
asso	9.	group size and source position

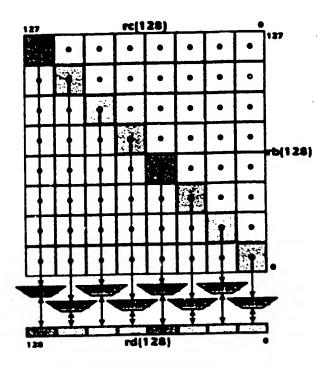
The 9-bit gssp field encodes both the group size, gsize, and source position, spos, according to the formula gssp = 512-4*gsize+spos. The group size, gsize, is a power of two in the range 1..128. The source position, spos, is in the range 0..(2*gsize)-1.

The values in the s, n, m, L and rnd fields have the following meaning:

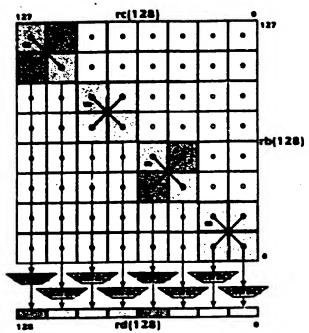
				1	rnd
0 1 2	unsigned signed	real complex	extract/same-sign merge/mixed-sign	truncate saturate	F Z N C

For the E.SCALADD.X instruction, bits 127..64 of the contents of register re specifies the multipliers for the multiplicands in registers ra and rb. Specifically, bits 64+2*gsize-1..64+gsize is the multiplier for the contents of register ra, and bits 64+gsize-1..64 is the multiplier for the contents of register rb.

An ensemble-multiply-extract-doublets instruction (E.MULX) multiplies vector ra [h g f e d c b a] with vector rb [p o n m l k j i], yielding the result [hp go fn em dl ck bj ai], rounded and limited as specified by rc31..0.

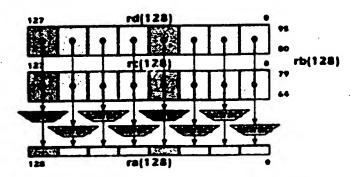


An ensemble-multiply-extract-doublets-complex instruction (E.MULX with n set) multiplies operand [h g f e d e b a] by operand [p o n m l k j i], yielding the result [pp+ho go-hp en+fm em-fn cl+dk ek-dl aj+bi ai-bj], rounded and limited as specified. Note that this instruction prefers an organization of complex numbers in which the real part is located to the right (lower precision) of the imaginary part.:



Ensemble complex multiply extract doublets

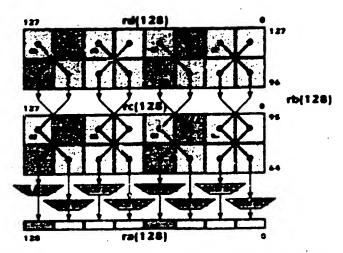
An ensemble-scale-add-extract-doublets instruction (E.SCAL-ADD.X) multiplies vector ra [h g f e d c b a] with re95..80 [r] and adds the product to the product of vector rb [p o n m l k 1 i] with rc79..61 [q], yielding the result [hr+pq gr+oq fr+nq er+mq dr+lq cr+kq br+jq ar+iq], rounded and limited as specified by rc31..0.



Ensemble scale add extract doublets

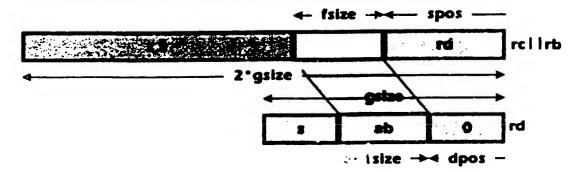
An ensemble-scale-add-extract-doublets-complex instruction (E.SCLADD.X with n set) multiplies vector ra [h g f e d c b a] with rc127.96 [t s] and adds the product to the product of vector rb [p o n m l k j i] with rc95..64 [r q], yielding the result [hs+gt+pq+or gs-ht+oq-pr

fs+et+nq+mr es-ft+mq-nr ds+ct+lq+kr cs-dt+kq-lr bs+at+jq+ir as-bt+iq-jr], rounded and limited as specified by re31..0.



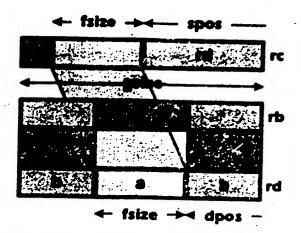
Ensemble complex scale add extract doublets

For the E.ENTRACT instruction, when m=0, the parameters are interpreted to select a fields from the catenated contents of registers rd and rc, extracting values which are catenated and placed in register rai:



Ensemble extract

For an ensemble-merge-extract (G.X when m=1), the parameters are interpreted to merge a fields from the contents of register rd with the contents of register rc. The results are catenated and placed in register ra.



Ensemble merge extract

Definition

signed ← b14

```
def mulisize,h,vs,v,i,ws,w,j) as
     mul ← ((vs&vsze-1+gh-size | | vsize-1+i, d * ((ws&wsize-1+gh-size | | wsize-1+j, d
enddef
def EnsembleExtract(op,ra,rb,rc,rd) as
     d ← RegRead(rd, 128)
     c ← RegRead(rc, 128)
     b ← RegRead(rb. 128)
     case b<sub>8.0</sub> of
          C..255:
               sgsize ← 128
          256..383:
               sgsize ← 64
          384..447:
               sqsize ← 32
          448..479:
               sgsize ← 16
          480..495:
               squze ← 8
         496..503:
               sgsize ← 4
         504..507:
               sgsize ← 2
         508.511:
              sqsize ← 1
    endcase
    1 - b11
    m ← b12
    n ← b13
```

.;

```
case op of
         E.EXTRACT:
               gsize ← sgsize
               h \leftarrow (2 \cdot m)^{\circ}gsize
               as ← signed
               spos \leftarrow (b<sub>8..</sub>r; and ((2-m)*gsize-1)
        E.SCALADD.X:
              if (sgsue < 8) then
                    gsize \leftarrow 8
              elseif (sg: vi -*(n+1) > 32) then
                    gsize \leftarrow 32/(n+1)
                    gsize ← sqsize
              endif
              ds \leftarrow cs \leftarrow signed
              bs ← signed * m
              as'← signed or m or n
              h \leftarrow (2^{\circ}gsze) + 1 + n
             spos \leftarrow (b<sub>8.0</sub>) and (2°gsize-1)
       E.MULX:
             if (sgsize < 8) then
                   gsize 4. 8
             elseif (sgsizer(:1+1) > 128) then
                   gsize ← 128/(n+1)
             else
             gsize ← sgsize
endif
             ds \leftarrow signed
             cs ← signed * m:
             as ← signed or m or n
             h \leftarrow (2^{\circ}gsize) + n
             spos ← lbe of and (2°gsize-1)
endcase
dpos ← (0 11 b23 16) and (gsize-1)
r \leftarrow spos
structure of 0.11 b_{31...24} and [gsize-1]
fisize ← (sfisize = 0) or ((sfisize+dpos) > gisze) ? gisze-dpos : sfisize
fsize - (tfsize + spos > h) ? h - spos : dsize
if |D_{10}| = |Z| and not as then
      rnd ← F
else
     rnd ← b10 9
endif
for i \leftarrow 0 to 128-gsize by gsize
     case op of
           E.EXTRACT:
                 if m then
                       p \leftarrow d_{gsize-1..1}
                       p \leftarrow |d| |1| |c|_{2}
                 endif
           E.MULX:
                 if n then
```

```
if ||\cdot| and ||gsze|| = 0 then
                             p - muligsize,h,dx,d,i.cx,c,i) - muligsize,h,dx,d,+size,cx,c,i+sizej
                             p - muligsize.h,ds.d,i.cs.c,+size| + muligsize,h,ds.d,i.cs.c,i+size|
                        endif
                  else
                       p ← muligsize,h,ds,d,i,cs,c,ij
                  endif
            E.SCALADD.X:
                  if n then
                        if || and g_{a}| = 0 then
                             p \leftarrow muligsize,h,ds,d,i,bs,b,64+2*gsizej
                                   + mul(gsize,h,cs,c,i,bs,b,64)
                                   - mul(gsze,h,ds,d,i+gsze,bs,b,64+3*gsize)
                                   - muligsize,h,cs,c,i+gsize,bs,b,64+gsize)
                       etse
                             p 		mullgsize,h,ds,d,i,bs,p,64+3*gsizel
                                   mul(gsize,h.cs,c,i,bs,b,64+gsize)
                                   + muligsize,h,ds,d,i+gsize,bs,b,64+2*gsize|
                                   + mulgsize,h,cs,c,i+gsize,bs.b.64)
                       endif
                 eise
                       endcase
      case raid of
           N:
                 s \leftarrow 0^{h-r} | 1 - p_r | 1 | p_r^{r-1}
           Z:
                 s -- Oh-1 11 ph-1
           F:
                 s \leftarrow 0^h
           C:
                 s - Oher II Ir
     endcase
     v \leftarrow (las & p_{h-1}) | p_l + (0) | s_l
      If |V_{n}|_{r+tyze} = |as & v_{r+tyze-1}|^{h+1-r-tyze}| or not (I and |op| = E.EXTRACT) then
           w - las & vietuze-1)guze-fuze-dpos | | Viuze-1+r r | | Odpos
     else
           W \leftarrow (s.7 \text{ Np. } 11 \text{ -Vg/size-dpos-1}) : 19 \text{size-dpos-} 11 \text{ Odpos}
     endil
     if m and lop = E.EXTRACT) then
           a_{\text{size-1+1}} \leftarrow c_{\text{gsize-1+1}, \text{dpos-fsize+1}} \cup c_{\text{dpos-1+1}}
           Size-14.1 ← M
     endr
endfor
RegWrite(rs, 128, 4)
```

Exceptions

700 470

mine

enude.

Ensemble Extract Immediate

These operations take operands from two registers and a short immediate value, perform operations on partitions of hits in the operands, and place the concatenated results in a third register.

Operation codes

E.EXTRACT.I.8.C	Eriser-ble entract immediate signed bytes criting
E.EXTRACT.I.8.F	Ensemble extract immediate signed bytes floor
E.EXTRACT.I.8.N	Ememble extra i immediate righed bytes nearest
E.EXTRACT.I.8 Z	Ensemble extract immediate signed bytes zero
E.FXTRACT.I.16.C	Ememble extract immodiate signed coubling criting
E.EXTRACT.I.16.F	Ensemble extract immediate signed doublets floor
E.EXTRACT. 16.N	Ensemble extract immediate signed doublets new est
E.EXTRACT.I.16.Z	Emerible cutact immediate signed doublets zero
E.EXTRACT.I.32.C	Exemble extract immediate signed quadlets ceiling
E.EXTRACT 32.F	Ememble extract immediate signed quartiess floor
E.EXTRACT.I.32.N	Ensemble extract immediate signed quadlets nearest
E.EXTRACT 1_2.Z	Enamble extract immediate signed quadlets zero
E.EXTRACT.I.64.C	En, emble extract immediate signed ucties ceiling
E.EXTRACT.1.64.F	Ememble extract immediate signed octiet: floor
E.EXTRACT.I.64.N	Ensemble extract invinediate signed octiets nearest
E.EXTRACT.1.64.Z	Ensemble extract immediate signed octiets zero
E.EXTRACIALUA C	Entertible extract immediate unsigned bytes ceiling
E.EXTRACT.I.U.8.F	Ensimble extract immediate unsigned bytes floor
E EXTRACT.I.U.8.N	Ensemble extract immediate unsigned bytes nearest
E.EXTRACT.I.U. 16.C	Ememble extract immediate unsigned doubless ceiling
E.EXTRACT.I.U. 1 o.F	Ensemble extract immediate unsigned doublets floor
E.EXTRACT.I.U. 16.N	Ensemble entract immediate imagned doublets neares:
E.EXTRACT.I.U.32.C	Ensemble extract immediate unsigned quadlets ceiling
E.EXTRACT.I.U.32.F	Ensemble extract immediate unsigned quadlets floor
E.EXIFACT.I.U.32.N	Ensemble extract immediate unsigned quadrets nearest
E.EXTRACT.I.U.64.C	Ensemble extract invitediate ransigned octiets ceiling
E.EXTRAC U.64.F	Ensemble entract immediate unsigned octies finds
E.EXTRACT.I.U.64.N	Ensemble eithe :: immediate umigned octiess nearest
E.MUL.X.I.8.C	Emericile multiply extract immediate signed bytes cealing
E.MULX.I.B.F	Ememble multiply extract immediate signed bytes floor
E.MUL.X.I.B.N	Ensemble multiply extract immediate signed bytes newest
E.MULX.I.8.Z	Ensemble multiply extract enrichate signed bytes zero
E.MUL.X.I. 16.C	Ensemble inumply extract immediate signed doublets ceaning
E.MUL.X.I.16.F	Emerable multiply extract immediate signed doublets floor
E.MUL.X.I.16.N	Emeritive multiply extract immediate signed coubics nearest
E.MUL.X.I. 16.Z	Ensemble multiply extract immediate signed rioublets rero
E.MUL.X.I.32.C	Ensective multiply extract immediate signed quadlets cosing
E.MUL.X.I.32.F	Emirror multiply estract immediate sujned quadlets floor